

● 陈志杰 赵书钦 李树钧 万福永 编

L^AT_EX

入门与提高 (第二版)



高等教育出版社
HIGHER EDUCATION PRESS

图书在版编目 (CIP) 数据

LATEX 入门与提高 / 陈志杰等编. —2 版. —北京: 高等教育出版社, 2006. 5 (2010 重印)

ISBN 978 - 7 - 04 - 019379 - 4

I. L… II. 陈… III. 排版 - 应用软件, LATEX
IV. TS803. 23

中国版本图书馆 CIP 数据核字 (2006) 第 045830 号

策划编辑 徐 可 责任编辑 张耀明
封面设计 王凌波 责任印制 张泽业

出版发行 高等教育出版社
社 址 北京市西城区德外大街 4 号
邮政编码 100120

经 销 蓝色畅想图书发行有限公司
印 刷 北京地质印刷厂

开 本 787 × 960 1/16
印 张 29.75
字 数 550 000

购书热线 010 - 58581118
免费咨询 800 - 810 - 0598
网 址 <http://www.hep.edu.cn>
<http://www.hep.com.cn>
网上订购 <http://www.landaco.com>
<http://www.landaco.com.cn>
畅想教育 <http://www.widedu.com>

版 次 2000 年 8 月第 1 版
2006 年 5 月第 2 版
印 次 2010 年 11 月第 6 次印刷
定 价 47.20 元 (含光盘)

本书如有缺页、倒页、脱页等质量问题, 请到所购图书销售部门联系调换。

版权所有 侵权必究

物料号 19379 - 00

第二版前言

本书出版3年以来受到广大读者的欢迎,考虑到 \LaTeX 的迅速发展,我们决定推出第二版.这里着重介绍第二版与第一版的差别.

首先我们改以CJK作为处理汉字的主线,不再介绍天元软件与CCT.我们觉得现在CJK已经足够成熟,与新开发的宏包兼容性也很好,应该是未来处理中文的主线.我们将停止开发天元软件,只作必要的维护,也不再鼓励新用户使用天元软件.如果读者想把天元文稿转换成CJK文稿,可以参看第二章第9节.

其次,我们重新组织开发人员,制作了基于MiKTeX Direct CD的随书光盘.我们本着“以用户为中心”的思路进行设计,精心制作.在随书光盘的开发过程中,我们收集了相关内容的最新版本,在多种Windows平台上进行了反复的调试,通过与相关产品开发者的紧密合作,解决了一些已有的bug,扩展了部分功能(如 \TeX 4ht的直接输出GBK字符的功能).这里也要感谢被收录在光盘内的宝贵资料的作者,你们的经验将被大家共享,这也是我们 \TeX 爱好者的共同愿望.

此外,第二版增加了绘图宏包的内容,并介绍了用 \LaTeX 生成演示文稿的beamer宏包以及生成html网页的 \LaTeX 2HTML.由于投影仪的广泛使用,而PowerPoint对科学公式的支持很差,因此为科学讲演制作演示文稿是一个使人头痛的问题.现在好了,使用 \LaTeX 及beamer宏包完全能制作出与PowerPoint媲美的演示材料,而且对于熟悉 \TeX 的人来说,不需学习太多的新知识.随着网上教学的普及,用 \LaTeX 制作教辅材料的需求也会越来越大,我们相信 \LaTeX 2HTML或 \TeX 4ht的用户也会不断增加.我们还增加了几个附录,包括 \LaTeX 宏包简介以及 \LaTeX 能使用的符号表,以便读者查阅.

\TeX 还在不断发展,希望我们的书能提供基本稳定的知识,而更新的内容就要依靠网络了.国内 \TeX 爱好者最常去的网站有:

CTeX (<http://www.ctex.org>)

ChinaTeX (<http://www.chinatex.org>)

希望读者有空常去看看,如果我们的随书光盘(实际上就是ChinaTeX CD)需要更新,也会发布在李树钧的个人TeX页面(<http://www.hooklee.com/tex.html>)以及ChinaTeX、CTeX网站上.

2004 年 7 月在华东师范大学召开的“中文 T_EX 与数学网站交流会”(<http://wims.ecnu.edu.cn/texmeeting>)是 T_EX 积极分子会面的盛会. 天元创始人肖刚、CCT 创始人张林波以及上述网站的掌门人都参加了. 如果大家想一睹他们的庐山真面目, 不妨去参观一下会议的主页.

陈志杰 (zjchen@math.ecnu.edu.cn)

赵书钦 (sqzhao@math.ecnu.edu.cn)

李树钧 (hooklee@hooklee.com)

万福永 (fywan@math.ecnu.edu.cn)

2005 年 12 月

第一版前言

我们从20世纪80年代后期开始认识和使用 $\text{T}_{\text{E}}\text{X}$,不久肖刚就开发了中文 $\text{T}_{\text{E}}\text{X}$ (天元软件的前身). 尽管当时在8088 PC机上运行 $\text{T}_{\text{E}}\text{X}$ 宛如牛步,从屏幕上看到“[”出现后还要等好一会儿才能看到另一半“]”的再现,真像在考验人的耐心. 可是那美丽的数学公式使我们马上丢掉了WordStar. 对于飞速发展的电子科技来说,十余年的时间相当于好几代:从8088、286、386、486、奔腾一二三代到奔腾四代,现在用 $\text{T}_{\text{E}}\text{X}$ 编译一本厚书也用不了几秒;软件方面同样如此:现在的年轻人已不知道WordStar、CCDOS为何物了. 可是 $\text{T}_{\text{E}}\text{X}$ 却经历了时间的考验,生存了下来,而且还在不断发展,成为网络时代发布和交流数学思想的重要工具. 这无疑应归功于创始人D. E. Knuth的远见和它的公开性. 当然 $\text{T}_{\text{E}}\text{X}$ 也在发展,目前,16位的 $\text{e}_{\text{M}}\text{T}_{\text{E}}\text{X}$ 早已不再更新,代之以32位的 $\text{M}_{\text{I}}\text{K}_{\text{T}}\text{E}_{\text{X}}$ 和 $\text{f}_{\text{P}}\text{T}_{\text{E}}\text{X}$, $\text{A}_{\text{M}}\text{S}-\text{T}_{\text{E}}\text{X}$ 也不再更新,它的许多功能都被 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 吸纳,成为 $\text{A}_{\text{M}}\text{S}-\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$. 有鉴于此,本书只介绍 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$,而且以 $\text{M}_{\text{I}}\text{K}_{\text{T}}\text{E}_{\text{X}}$ 为默认的系统.

根据作者自己的体验,软件用户一般没有耐心去阅读厚厚的说明书,他们只希望知道如何快速安装,再学习一点最基本的规则,然后就模仿样板例子自己去闯,等遇到问题时再去查说明书. 本书就是针对用户的这个特点编排的. 全书分成“基本篇”和“提高篇”,基本篇面向初学者,按教材的规格编写,学完此篇后就能用 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 打印自己的数学论文. 提高篇是供选读或需要时查阅的,因此各部分相对独立,并不遵照线性的次序. 为了方便读者查找自己需要的内容,我们把小节名也列入目录,书末还有 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 的命令简介以及详细的索引. 此外,本书还附有光盘,其中包含了3套 $\text{T}_{\text{E}}\text{X}$ 系统:第一种方案是 $\text{M}_{\text{I}}\text{K}_{\text{T}}\text{E}_{\text{X}}$ 以及天元的完整安装;第二种方案只在硬盘上安装少量辅助文件,在光盘上运行 $\text{T}_{\text{E}}\text{X}$;这两种方案都能实现本书讲解的所有功能. 第三种方案是面向机器配置较低,无法实现第一种方案的用户,它可以在DOS环境下运行,不过它不能保证本书中某些内容的实现. 附录一专门讲解如何利用光盘安装 $\text{T}_{\text{E}}\text{X}$ 系统. 光盘上还包括了本书的许多实例,可供读者参考.

为了使 $\text{T}_{\text{E}}\text{X}$ 也能处理汉字,本书以天元预处理程序作为主线介绍,在第十二章也介绍了CJK宏包,这是一个很有前景的中文解决方案,只是由于使用经验尚少,对于它的兼容性没有把握,因此没有作为本书的主线. 对于另一个中文预处理程序CCT,我们也在第十二章作了介绍.

随着传播和展示手段的不断更新, $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 的输出方式也趋向多样化, 除了传统的输出到纸质媒体上外, 也可以通过电脑屏幕或投影片的中介输出到投影仪上, 为使附有图形的论文能够传输给同行, 可以输出成 pdf 或 ps 格式, 还可以把 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 的源文件直接输出成 html 格式, 以便在网上公布. 所有这些, 通过 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 的扩展, 都是可以实现的, 我们在第十三章对可以使用的方法作了介绍.

本书在写作过程中得到了上海市重点学科建设项目的资助, 北京师范大学的何青教授在审稿过程中提出了宝贵的意见和建议, 在此表示衷心感谢. 我们还要感谢许许多多 $\text{T}_{\text{E}}\text{X}$ 朋友, 他们向我们提供的经验和信息对提高本书的质量起了很大作用. 由于作者的经验有限, 许多内容是现学现写的, 错误之处一定不少, 恳请各位读者不吝指教. 欢迎大家与我们联系, 也欢迎 $\text{T}_{\text{E}}\text{X}$ 之友有兴趣时到华东师范大学数学系的网站 (<http://www.math.ecnu.edu.cn>) 上看看, 其中有关天元软件的更新消息, 也许我们在有条件时会开一个 $\text{T}_{\text{E}}\text{X}$ 的交流点.

陈志杰 (zjchen@math.ecnu.edu.cn)

赵书钦 (sqzhao@math.ecnu.edu.cn)

万福永 (fywan@math.ecnu.edu.cn)

2001 年 12 月

目 录

基 本 篇

第一章 引言	3
§1.1 \TeX 和 \LaTeX	3
§1.2 天元软件及 CCT 系统	5
§1.3 CJK 宏包	5
§1.4 排版过程	5
§1.5 \TeX 中的长度	6
1.5.1 固定长度	6
1.5.2 弹性长度	7
第二章 准备文稿	8
§2.1 基本格式	8
2.1.1 纯西文源文件基本格式	8
2.1.2 含有中文的源文件基本格式	9
2.1.3 自定义页芯大小	11
§2.2 输入特殊字符	11
2.2.1 几个特殊字符	11
2.2.2 空格与换行	12
2.2.3 引号、连字号、破折号	12
2.2.4 连体字	13
2.2.5 句号后的空白	13
2.2.6 非英文字母及重音符号	14
§2.3 分组与环境	14
§2.4 分段	15
§2.5 分行和分页	15
2.5.1 分行	15
2.5.2 分页	16
§2.6 水平间距、竖直间距	17
2.6.1 水平间距	17
2.6.2 导引线	19

2.6.3 竖直间距	19
§ 2.7 与段落有关的距离	20
2.7.1 首行缩进	20
2.7.2 段落间距	21
2.7.3 伸展行距	21
§ 2.8 CJK 小结	21
§ 2.9 天元文稿转换成CJK文稿	25
2.9.1 增删一些语句	25
2.9.2 转换字体命令	25
2.9.3 转换字体尺寸命令	26
2.9.4 转换字形命令	26
第三章 文字模式	27
§ 3.1 西文字体	27
3.1.1 字体属性	27
3.1.2 选择字体尺寸	28
§ 3.2 中文字体	30
§ 3.3 居中	32
§ 3.4 参考文献	33
§ 3.5 制表位	34
§ 3.6 表格	36
3.6.1 表格环境	36
3.6.2 样例	38
§ 3.7 脚注	40
第四章 数学公式	41
§ 4.1 概述	41
§ 4.2 行内公式	43
§ 4.3 行间公式	43
§ 4.4 上标和下标	45
§ 4.5 分式	46
§ 4.6 根式	48
§ 4.7 求和、积分	50
§ 4.8 数学重音符号	51
§ 4.9 上画线、下画线及类似符号	52
§ 4.10 堆叠符号	53
§ 4.11 可以变大的定界符	55
§ 4.12 矩阵	56
§ 4.13 单行公式与多行公式	58
§ 4.14 数学字体	62

4.14.1 选择数学字体	62
4.14.2 希腊字母	63
4.14.3 数学粗体	64
§ 4.15 数学符号表	65
4.15.1 二元运算符	65
4.15.2 关系运算符	65
4.15.3 箭头符号	66
4.15.4 其他符号	67
4.15.5 具有两种尺寸的符号	67
4.15.6 函数名	67
第五章 常用文档的类别与版式	69
§ 5.1 文档类别命令中的可选项	69
5.1.1 指定基本字体尺寸	69
5.1.2 指定纸张大小	70
5.1.3 其他一些选项	70
§ 5.2 章节	70
§ 5.3 文章标题	73
§ 5.4 摘要	74
第六章 图形	77
§ 6.1 图形与坐标系	77
§ 6.2 基本绘图命令	78
6.2.1 直线和矢量线	78
6.2.2 圆和圆角矩形	80
6.2.3 图形中的盒子	81
6.2.4 图形中的文本	83
6.2.5 曲线	84
§ 6.3 子图	84
§ 6.4 天元的绘图功能	89
§ 6.5 浮动表格和图形	94
第七章 自定义与改错	97
§ 7.1 自定义命令	97
§ 7.2 给计数器和长度赋值	99
7.2.1 计数器	99
7.2.2 长度	100
§ 7.3 出错信息	101

§ 7.4 警告信息	103
------------------	-----

提 高 篇

第八章 文字模式的高级技巧	107
§ 8.1 使文本居左或居右	107
§ 8.2 引文	108
§ 8.3 抄录	108
§ 8.4 盒子	109
8.4.1 LR 盒子	110
8.4.2 LR 盒子的升降	112
8.4.3 标尺盒子	113
8.4.4 子段盒子与小页环境	113
§ 8.5 制表位的高级技巧	114
§ 8.6 表格的高级技巧	115
8.6.1 更多的表格参数	115
8.6.2 几个表格样例	117
§ 8.7 罗列	120
8.7.1 三种罗列环境	120
8.7.2 改变默认的罗列条目标签	121
§ 8.8 广义罗列环境	123
8.8.1 标准标签	123
8.8.2 广义罗列环境的样式参数	124
8.8.3 平凡罗列环境	126
§ 8.9 脚注与边注	126
8.9.1 自动编号的脚注	127
8.9.2 指定编号的脚注	127
8.9.3 禁止模式中的脚注	127
8.9.4 边注	128
§ 8.10 段落形状	129
8.10.1 移动段落边界	129
8.10.2 多行缩进	130
8.10.3 段落形状命令	130
第九章 数学公式排版的一些技巧	132
§ 9.1 数学排版的国际标准	132
§ 9.2 数学模式中的字体尺寸	133
§ 9.3 数学模式中的参数	134
§ 9.4 定理定义的排版	135

§ 9.5 巧妙使用阵列环境	137
§ 9.6 多行公式左列问题	140
§ 9.7 amsmath 宏包简介	140
§ 9.8 公式中的文本 (amsmath)	142
§ 9.9 单个公式 (amsmath)	143
§ 9.10 方程组 (amsmath)	145
9.10.1 gather 环境	145
9.10.2 align 环境	146
9.10.3 flalign 环境	147
9.10.4 alignat 环境	147
9.10.5 gathered, aligned 和 alignedat 环境	148
9.10.6 cases 环境	149
§ 9.11 矩阵 (amsmath)	150
§ 9.12 多重数学符号 (amsmath)	151
9.12.1 多重角标	151
9.12.2 多重积分	152
9.12.3 叠置重音符号	152
9.12.4 省略号	153
§ 9.13 分式 (amsmath)	154
9.13.1 普通分式	154
9.13.2 连分式	154
9.13.3 二项式系数	155
9.13.4 自定义分式类命令	155
§ 9.14 函数 (算子) 名 (amsmath)	156
9.14.1 已定义的函数名	156
9.14.2 定义新的函数名	157
§ 9.15 其他功能 (amsmath)	157
9.15.1 公式中的空白间隔	157
9.15.2 调整根式指数的位置	158
9.15.3 调整公式编号的竖直位置	158
9.15.4 特殊的上下标 (上下限)	159
9.15.5 不可断行的区间符	160
第十章 新字体选择方案 (NFSS)	161
§ 10.1 NFSS 中的字体属性	161
§ 10.2 简化的字体选择命令	163
§ 10.3 属性的默认值	165
§ 10.4 定义新的字体命令	166
§ 10.5 定义新的数学字体命令	166
10.5.1 数学字母字体	166

10.5.2 数学符号字体	167
§ 10.6 在 NFSS 下指定字体	169
§ 10.7 编码命令	172
§ 10.8 计算机现代字体	173
10.8.1 简介	173
10.8.2 使用 CM 字体	174
§ 10.9 ttshape 简介	175
10.9.1 改变字形	176
10.9.2 pdfL ^A T _E X、dvipdfmx、dvips 补充命令	176
10.9.3 直立希腊字母	178
第十一章 文档的布局及相互联系	179
§ 11.1 分栏	179
§ 11.2 单、双面	180
§ 11.3 与公式有关的选项	181
§ 11.4 页版式	182
11.4.1 页面布局	182
11.4.2 指定页眉内容	184
11.4.3 页码	184
§ 11.5 自定义页眉和页脚	186
11.5.1 一个简单例子	186
11.5.2 使用章节标题作页眉	187
§ 11.6 目录表及图表清单	188
§ 11.7 文档的分割处理	190
11.7.1 \input 命令	190
11.7.2 \include 命令	190
11.7.3 人机交互命令	192
§ 11.8 交叉引用	193
11.8.1 交叉引用	193
11.8.2 索引记录	194
第十二章 图形包介绍	196
§ 12.1 图形包 graphics	197
12.1.1 插入图形的基本命令	199
12.1.2 放大和缩小	199
12.1.3 水平翻转和旋转	201
§ 12.2 用宏包 amscd 画交换图	203
§ 12.3 用宏包 diagrams 画交换图	205
§ 12.4 Xy-pic	210
§ 12.5 MetaPost	220

§ 12.6 PSTricks	236
§ 12.7 PGF	253
§ 12.8 其他绘图宏包	262
第十三章 输出到投影仪或互联网	264
§ 13.1 如何显示彩色	264
§ 13.2 如何使用 pdfTeX	266
§ 13.3 pdfscreen 与 pdfslide	271
13.3.1 用 pdfscreen 生成适合屏幕阅读的 pdf 文件	271
13.3.2 用 pdfslide 生成演示用 pdf 文件	272
§ 13.4 用 beamer 生成演示用 pdf 文件	273
13.4.1 快速入门	274
13.4.2 高级技巧	280
13.4.3 整体设计——主题	288
§ 13.5 用 slides 制作透明片	288
§ 13.6 用 L ^A T _E X2HTML 生成 html 文件	291
§ 13.7 用 T _E X4ht 生成 html 文件	298
附录一 T_EX 系统的安装	301
A1.1 快速安装	301
A1.2 安装测试	303
A1.3 手工安装	306
A1.4 如何制作自己的 MiKTeX Direct	308
A1.5 转换为 MiKTeX Direct 工作模式	309
附录二 L^AT_EX 命令简介	310
附录三 字体表	371
附录四 L^AT_EX 的宏包	379
附录五 L^AT_EX 能使用的符号	406
参考文献与网站	440
索引	441

基 本 篇

第一章 引言

计算机的发展带动了各行各业的发展,使很多行业出现了革命性的变化,例如印刷出版业现已告别铅与火的时代,普遍使用计算机排版系统.

在计算机排版系统出现之前,人们发表文章或出版书籍时是作者将手稿提供给编辑部或出版社,由专职编辑人员在手稿上作文字修改并添加排版指令,交排版工人排出校样,由作者校对后再返回编辑重复上述过程,一般要重复几次,每次重复还有可能出现新的排版错误.对排好的校样,如果要更改版面设置,就需要重排,工作量是很大的.有了计算机排版系统,情况就大不相同了,录入人员(或作者本人)把原稿输入计算机,编辑人员添加排版指令后,可以直接输出用于印刷的胶片.改变字体、版面等设置是很简单的操作.

目前,世界上已经有许多大大小小的排版系统,各有其特点和适用范围,例如方正电子出版系统已是国内大多数报社的首选系统,而普通用户在编排要求不高的稿件时,使用所见即所得的 Word、WPS 等软件也不失为合适的选择.

本书介绍的 $\text{T}_\text{E}\text{X}$ 系统是一种使用方便、价格低廉的排版软件,当排版论文、报告和书籍时,其输出质量并不逊色于价格昂贵的大型系统,在某些方面(例如排版数学公式)仍是排版质量最好的系统.

§ 1.1 $\text{T}_\text{E}\text{X}$ 和 $\text{L}_\text{A}\text{T}_\text{E}\text{X}$

$\text{T}_\text{E}\text{X}$ 系统是由美国 Stanford 大学教授 Donald E. Knuth 研制的计算机排版软件系统. Knuth 有一个中文名字——高德纳,他在国际上既是著名的数学家,又是著名的计算机专家,是享有盛誉的计算机程序设计系列专著 The Art of Computer Programming 的作者. 按着他的计划,这套书总共六册. 前三册出版后,Knuth 将修订的第二册第二版手稿交出版社排版,但对收到的校样很不满意,因为这时出版社已开始用计算机代替手工排版,当时的字形和版面都很难看,每次的校改也非常麻烦. 为了以后出书的方便,他放下手头的工作,开始设计一套高质量的计算机排版软件,花费大量的精力和时间后,研制成功了这套闻名于世的 $\text{T}_\text{E}\text{X}$ 系统. 稍后他又撰写了一整套 $\text{T}_\text{E}\text{X}$ 手册,既讲 $\text{T}_\text{E}\text{X}$ 的使用方法,又讲设计原理,这套书也成了享有盛誉的经典之作. 更难能可贵的是他并没有利用 $\text{T}_\text{E}\text{X}$ 系统去发财致富,而是无私地把源代码向用户公开.

Knuth 为其研制的软件命名为 $\text{T}_\text{E}\text{X}$, 取意于希腊词根 $\tau\epsilon\chi$, 因此名称中的 X 应读

χ 的音, 即 $\text{T}_{\text{E}}\text{X}$ 的发音为 [tex] ([x] 的发音类似于汉语拼音的 h) 或 [tek] 而不是 [teks], 这也使得该软件的名称在外形和读音上都不同于另一个软件 TEX . 在纯文本环境中, 通常将 $\text{T}_{\text{E}}\text{X}$ 写成 TeX .

利用 $\text{T}_{\text{E}}\text{X}$ 系统编写手稿, 除了正文内容之外, 还需加入一些排版命令. 与大型排版系统不同的是, 这些排版命令通常不是编辑人员加入的, 而是由作者本人完成的.

$\text{T}_{\text{E}}\text{X}$ 提供的排版命令功能强大, 用户可以直接使用这些命令, 也可以发挥创造性, 利用已有的功能自行定义新的命令, 以适合特定的需要.

$\text{T}_{\text{E}}\text{X}$ 系统提供了 300 多条基本命令 (其中有很大部分是键盘上没有的特殊符号的代码), 功能虽然强大, 但使用不够方便. 后来在这些基本命令的基础上, 又定义了 600 多条复合命令, 构成名为 Plain $\text{T}_{\text{E}}\text{X}$ 的宏包 (软件包), 当人们专指 $\text{T}_{\text{E}}\text{X}$ 而不是衍生版本时, 实际指的就是 Plain $\text{T}_{\text{E}}\text{X}$.

Plain $\text{T}_{\text{E}}\text{X}$ 虽然比“原始系统”易用, 但排版复杂的版面或公式时仍需书写大量的命令, 还是不够方便, 因此国外许多人利用 $\text{T}_{\text{E}}\text{X}$ 的宏定义功能进行二次开发, 产生了一些 $\text{T}_{\text{E}}\text{X}$ 系统的衍生版本, 其中最著名的是由美国数学会 (AMS) 组织人员编写的 $\mathcal{A}\mathcal{M}\mathcal{S}\text{-T}_{\text{E}}\text{X}$ 和 Leslie Lamport 编写的 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. 前者的特点是容易排版复杂的数学公式, 后者适合于排版普通文章及书籍. 但若把两者的优点组合起来则更符合人们愿望, 于是又出现了兼容于 $\mathcal{A}\mathcal{M}\mathcal{S}\text{-T}_{\text{E}}\text{X}$ 而又包含了 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 优点的衍生版本, 但没有广泛流行, 倒是 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 由于在新版本 ($\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$) 中可以加载 amsmath 宏包, 基本包含了 $\mathcal{A}\mathcal{M}\mathcal{S}\text{-T}_{\text{E}}\text{X}$ 的优点而大为流行, 占据了 $\text{T}_{\text{E}}\text{X}$ 领域的重要位置, 所以本书重点介绍 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 系统, 后文中的 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 实际是指 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$.

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 系统的特色功能之一是它的自动编号功能. 文章、书籍的章、节、段落以及公式、图表、文献、页码等均可自动编号, 这给作者带来很大方便, 例如增添或删除一个带有编号的公式, 其他的文字不用任何修改, 所有编号都会自动改变, 对编号及其所在页码的引用也都会自动改变. $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 系统还可以自动生成目录页, 可以自动生成索引附录.

为行文方便, 本书用 $\text{T}_{\text{E}}\text{X}$ 泛指 Plain $\text{T}_{\text{E}}\text{X}$ 、 $\mathcal{A}\mathcal{M}\mathcal{S}\text{-T}_{\text{E}}\text{X}$ 和 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$.

$\text{T}_{\text{E}}\text{X}$ 排版系统出现后受到了科学工作者喜爱, 因为它能排出复杂的图表和精美的数学公式, 到目前为止, 国内外公认的数学公式排得最好的排版软件仍是 $\text{T}_{\text{E}}\text{X}$ 系统. 许多国际专业学会的期刊杂志都欢迎作者使用 $\text{T}_{\text{E}}\text{X}$ 系统投稿, 一些出版社也使用 $\text{T}_{\text{E}}\text{X}$ 系统出版书籍.

$\text{T}_{\text{E}}\text{X}$ 的输出文件是 DVI (DeVice Independent, 设备无关) 文件, 这种文件在针式打印机、喷墨打印机、激光打印机以及照排机上的输出版面是完全相同的, 仅仅是文字或图形的分辨率因设备的不同而有所区别.

随着因特网的急速发展, 已有软件可将 $\text{T}_{\text{E}}\text{X}$ 文件转换成 HTML 文件, 放到网页中通过浏览器阅读.

为了用户安装和使用的方便, 开发者将各种 $\text{T}_{\text{E}}\text{X}$ 系统包装起来并添加一些功能, 做成发行版本, 常见的有 MiKTeX 、 emTeX 、 PCTeX32 、 fpTeX 、 TeXTeX 等版本. 其中 PCTeX32 是商业软件, 其他是免费软件, 可从网上自由下载.

§ 1.2 天元软件及 CCT 系统

T_EX 是面向西文的排版系统, 不能直接用于中文, 为了充分利用 T_EX 系统的功能, 国内的一些学者开发了几种中文预处理系统, 只需用这些预处理系统处理一下含有中文的原稿, 就可以使用 T_EX 系统了. 人们通常把中文预处理系统与 T_EX 系统合称为中西文 T_EX 排版系统. 其中最有名的是 CCT 系统和天元系统, 两者都是免费软件, 都有广泛的人群在使用, 不过因为设计原理不完全相同, 因此并不兼容.

上述两个系统都是多年前开始开发的, 当时的计算机与现在比还是很落后的, 因此两者有一个共同点, 就是只生成所需汉字的点阵字库, 以加快处理速度并减少硬盘的占用量. 两个系统也在与时俱进, 但上述设计基础未变, 中文与西文的处理未能统一起来, 由此带来两点不便: 一是多了一个中文预处理过程, 二是生成的 dvi 文件不能离开当前环境, 不过第二点不是一个大问题, 因为可在当前环境下将 dvi 文件转换成 pdf 文件或 ps 文件, 就可与其他人交流了.

§ 1.3 CJK 宏包

使 L^AT_EX 能处理中文 (Chinese)、日文 (Japanese)、韩文 (Korean) 的 CJK 宏包是 Werner Lemberg 开发的, 目前发布的 4.5 版已相当成熟, 现在能够支持多种亚洲双字节文字. 与天元和 CCT 系统相比, CJK 与 L^AT_EX 结合得更为紧密, 对字体的定义遵循新字体选择方案 (NFSS), 生成的 dvi 文件是标准的设备无关文件, 脱离当前系统仍可使用, 便于国际交流. CJK 与天元系统一样可以使用多种字库, 如 TrueType 轮廓字库, 因而 CJK 的输出可以达到与天元一样的美观效果. 但 CJK 对硬盘容量的需求很大, 欲试用的读者应该有较高的硬件配置. 此外要注意 CJK 宏包需要 1998 年 6 月 1 日以后的 L^AT_EX 版本.

现在的计算机硬盘容量巨大, 安装和使用 CJK 不成问题. 鉴于 CJK 处理中西文的一致性以及它的国际性, 本修订版在讲解中文处理时, 将以 CJK 为主线, 本书就是利用 L^AT_EX + CJK 排版完成的.

§ 1.4 排版过程

排版过程主要有以下几个步骤:

1. 首先是编写或修改文稿 (源文件), 在源文件中插入排版命令, 具体的编写要求将在后面的章节中详细介绍. 需要提醒的是必须保证源文件是纯文本格式, 即除了作者直接输入的字符以外, 不能含有其他的東西 (例如控制字符). 如果你使用 T_EX 的集成环境 (如 WinEdt, WinShell, PCTeX32), 那么有内嵌的编辑软件可供使用, 否则, 有很多文本编辑软件, 如 DOS 环境下的 EDIT 和 Windows 环境下的 NOTEPAD (记事本) 都符合条件. 当使用编辑功能很强的软件如 Word、WPS 等时则要小心, 保存文件时必须选择纯文本格式. 习惯上源文件的扩展名用 tex.

2. 用 \LaTeX 软件处理 tex 文件, 产生 dvi 文件. 如果这一步出现错误, 则表明源文件中的某个或某些排版命令有错误, 应返回第一步进行修改.

3. 在屏幕上显示或在打印设备上输出 dvi 文件, 察看文件内容或排印的版面是否合乎要求, 如果有不满意之处, 返回第一步进行修改.

4. 上述几步通过后, 就可以正式打印输出或照排制版了.

在 Windows 操作系统下有好几种集成环境可以使用, 在 DOS 环境则可用命令行方式分步完成, 但较麻烦, 使用集成环境则简单得多. 附录一将重点介绍 MiKTeX 以及集成环境 WinEdt 或 WinShell 的安装方法.

§ 1.5 \TeX 中的长度

\TeX 中的长度可分成两类, 一类是固定长度, 一类是根据排版情况可以伸缩的弹性可变长度.

1.5.1 固定长度

\TeX 中固定的长度、宽度或距离用十进制小数和一个长度单位组成, 数字可以带有正负号, 小数点可采用英美方式(使用圆点句号), 也可采用欧洲方式(用逗号作小数点). 常用的单位有如下几种:

mm	毫米
cm	厘米, $1\text{ cm} = 10\text{ mm}$
in	英寸, $1\text{ in} = 2.54\text{ cm}$
pt	点, $1\text{ in} = 72.27\text{ pt}$
em	与当前字体尺寸有关, 相当于大写字母 M 的宽度.
ex	与当前字体尺寸有关, 相当于小写字母 x 的高度.

也可以使用下列较少用到的长度单位:

bp	大点, $1\text{ in} = 72\text{ bp}$
pc	pica, $1\text{ pc} = 12\text{ pt}$
dd	didot, $1157\text{ dd} = 1238\text{ pt}$
cc	cicero, $1\text{ cc} = 12\text{ dd}$
sp	scaled point, $1\text{ pt} = 65536\text{ sp}$, sp 是 \TeX 中的最小单位, \TeX 中的所有量度值都是 sp 的整数倍.

当长度为 0 时, 不能只写数字 0, 必须附上单位, 例如写成 0 mm 或 0 pt 等.

em , ex 与当前字体尺寸有关, 粗略地说 em 相当于当前字体大写字母 M 的宽度, 实际比字母 M 的宽度更大些, 例如当前字体为 10 pt 时, em 就是 10 pt , 而 M 的宽度略小于 10 pt .

1.5.2 弹性长度

所谓弹性长度就是根据排版需要可以自动伸长或缩短的长度, 这种长度实际上由 3 个非负的长度组成, 一是正常长度, 即没有伸缩时的长度, 二是伸长时可以增加的长度, 三是缩短时最多可以减少的长度. 定义弹性长度的语法是:

`正常值 plus 伸展值 minus 收缩值`

当几个弹性长度被同时拉伸或压缩时, 实际伸展或收缩值是根据各自的伸展值或收缩值按比例分配的, 实际伸展长度可以超过定义中的伸展值, 但压缩的长度最大只能是定义中的收缩值.

为了能自动排版出优美的版面, 在 T_EX 中使用了大量的弹性长度. 此外还有两个特殊的弹性长度 `\fil`, `\fill`, 它们的正常长度是零, 但可以伸展到任何长度, 两者的区别是强度不同, “l” 越多强度越大, 当强者和弱者遇到一起时, 弱者好像消失了, 只有强者起作用. 在系统内部使用这种特殊的弹性长度时, 通常用的是 `\fil`, 只要用户在外部使用了 `\fill`, 就取消了 `\fil` 的作用, 此时只有 `\fill` 起作用.

第二章 准备文稿

§ 2.1 基本格式

文稿(即用于排版的源文件)包含两部分内容:一部分是正文,也就是需要排版输出的内容;另一部分是排版控制命令,用于控制版面式样、字体字形等格式.控制命令是用倒斜线引导的字符串.

排版命令分为两种,一种是“控制字”,一种是“控制符”.控制字由倒斜线和一个或多个英文字母组成,区分大小写,可以用任何非字母的字符(例如空格、数字、汉字、括号、标点符号等)表示控制字的结束.控制符由倒斜线和一个符号(不是字母)组成.有一些排版命令带有参数,不可省略的参数放在花括号中,可以省略的参数(可选参数)放在方括号中,这种带参数的命令格式为

`\命令名[可选参数]{不可省略的参数}`

当同一个括号中含有多个参数时,需用逗号分隔参数.为了区分带参数和不带参数的命令,有时把某些(并非全部)不带参数的命令称为声明.

2.1.1 纯西文源文件基本格式

纯西文(不含中文)源文件基本格式如下所示:

```
\documentclass[11pt]{article}
\begin{document}
...
\end{document}
```

建议用tex做文件的扩展名,扩展名与主名之间用“.”分隔.

语句

```
\documentclass[11pt]{article}
```

用于决定文章的版式类别和基本字体的大小,花括号中的article表示这是一篇普通文章,若将它改为book,则表示排成书籍样式.不同的版式类别具有不同的默认样式.方括号中的11pt表示基本字体大小为11pt.方括号中也可改用10pt或

12pt. 若不指定基本字体大小, 则默认使用 10pt, 此时若方括号中无其他选项, 需省略方括号.

语句

```
\begin{document}
```

和

```
\end{document}
```

表示正文的开始和结束. 所有正文要写在这两句之间, 可以写多行.

在 L^AT_EX 中把

```
\begin{xxx}  
\end{xxx}
```

称作“xxx 环境”, 在原稿中必须有一个而且只能有一个 document 环境.

在语句 `\documentclass[11pt]{article}` 与 语句 `\begin{document}` 之间可以放一些全局控制命令, 这个区域通常被称为是“导言区”.

2.1.2 含有中文的源文件基本格式

当原稿正文中有汉字时, 需要把汉字放在 CJK 环境中, 而且还要在导言区引入 CJK 宏包, 整体结构示例如下:

```
\documentclass[11pt]{article}  
\usepackage{CJK}  
\begin{document}  
\begin{CJK}{GBK}{song}  
祝贺你, MiKTeX 和 CJK 安装成功了!  
\end{CJK}  
\end{document}
```

CJK 环境中的参数 GBK 表示使用的编码为扩展国标码 GBK 大字符集, 参数 song 表示默认使用宋体字, 其他可用的编码和字体参见 § 2.8 节. 将上面的例子命名为 2-1-1.tex (本书例子第一个数字表示章, 第二个数字表示节, 第三个数字表示该例在本节中的序号), 用 L^AT_EX 编译后输出结果是:

```
祝贺你, MiKTeX 和 CJK 安装成功了!
```

如果想把上例中的短语“MiKTeX 和 CJK”改为粗体, 只需用花括号将它们括起来组成一个“分组”(或称集团), 并对它们使用粗体命令, 见下例(参见 2-1-2.tex).

```

\documentclass[11pt]{article}
\usepackage{CJK}
\begin{document}
\begin{CJK}{GBK}{song}
祝贺你, \textbf{MiKTeX 和 CJK} 安装成功了! .% 注意花括号
\end{CJK}
\end{document}

```

带有参数的命令 `\textbf{...}` 表示参数用粗体字输出, 输出结果是:

```

祝贺你, MiKTeX 和 CJK 安装成功了!

```

以 % 开始的语句是注释内容, 从这个符号开始直到行末的所有字符以及下一行行首的空白字符均被 $\text{T}_{\text{E}}\text{X}$ 忽略, 注释内容对排版没有影响, 也不会产生输出。% 号除表示注释外, 还有一个重要功能, 就是当文稿分行输入时, 如果行尾的空格或回车影响了排版时, 可在该行行尾加上 % 号。虽然大多数情况下文稿行尾不必加 %, 但若在命令的两个参数间分了行而又要避免行尾隐含的空格时, 就必须在行尾加 % 了。

$\text{T}_{\text{E}}\text{X}$ 源文件是一种自由格式文件, 输入源文件时不必考虑每行的长短, 也不必考虑单词之间空白的多少, 只要符合下面几节的一些输入要求, $\text{T}_{\text{E}}\text{X}$ 系统就会自动按着你的命令排版。因此建议在输入源文件时, 为了便于校对和改错, 每行不要太长, 最好在标点符号后面换行, 也不要忘了在西文标点后面应空一格。各种排版环境的开始命令和结束命令最好是单独占一行, 就像前面的例子一样。

CJK 宏包定义了两种环境, 其一是此处介绍的 CJK 环境, 该环境不会自动忽略汉字后面的空格, 因此在输入原稿时, 通常不要在汉字之间留有空格, 当原稿一个句子很长需要换行输入时, 应在换行的行末加一个 %, 以去掉换行产生的空格。

另一种是 CJK* 环境 (多了一个 * 号), 该环境自动忽略汉字后面的空格, 即汉字后的空格对排版不起作用, 如果需要排版后在某个汉字后面有空格时 (例如汉字后面是西文), 需在输入原稿时加入 “受保护的空格”, 即输入 `_` 或符号 `~`, 其中 `_` 表示在倒斜线后紧跟一个空格。

实际的空格是看不见的, 人们在板书或文章中常常用符号 “`□`” 表示空格, 本书也是如此, 但只对原稿中需要特别注意的空格才使用这个符号。

使用 CJK* 环境, 第一个例子应写成 (见 2-1-3.tex):

```

\documentclass[11pt]{article}
\usepackage{CJK}
\begin{document}
\begin{CJK*}{GBK}{song}
祝贺你, MiKTeX\_和\_CJK\_安装成功了!
\end{CJK*}
\end{document}

```

在中西文之间输入空格(CJK 环境)或受保护的空格(CJK* 环境), 排版输出后显得间距稍大, 不用空格而改用一个间隔“\,”(见第 17 页)较好, 或者在 CJK* 环境使用符号“~”. 有关 CJK 更多的一些知识, 参见 §2.8 节.

2.1.3 自定义页芯大小

用户如果不想使用 TeX 系统自动确定的页芯大小(参见第 70 页), 可以直接指定页芯大小, 只需在导言区写上命令

```
\setlength{\textwidth}{页芯宽度}
\setlength{\textheight}{页芯高度}
```

其中的宽度和高度可以使用 mm, cm 或 in 作长度单位. 注意页芯宽度不含边注, 页芯高度不含页眉和页脚.

打印时默认在纸张顶端和左侧留有一英寸(约 25.4 mm)的空白, 用户可以加大或缩小这种空白. 命令

```
\setlength{\voffset}{长度}
\setlength{\hoffset}{长度}
```

分别用于调整顶端和左侧的空白宽度, 参数长度为正值时增大空白宽度, 为负值时缩小空白宽度. 例如在导言区写上

```
\setlength{\voffset}{-15.4mm}
\setlength{\hoffset}{10mm}
```

打印时顶端与左侧的空白宽度分别变为 10 mm 和 35.4 mm.

§ 2.2 输入特殊字符

2.2.1 几个特殊字符

大部分键盘字符都可直接输入, 但字符“# \$ % { } ~ _ ^ \ | < >”在 TeX 中有特殊用途, 如果需要排版输出前面 9 个字符, 可按下表对应输入:

输出字符	#	\$	%	{	}	~	_	^	\
输入序列	\#	\\$	\%	\{	\}	\~{ }	\- { }	\^ { }	\$\backslash\$

接下去的 3 个字符“| < >”如果直接输入, 其打印结果成了“— ¡ ¿”, 完全变了样. 为得到正确的输出, 应分别输入成 \textbar \textless \textgreater, 或将它们用美元号 \$ 括起来, 即输入“\$| \$ \$< \$ \$>\$”, 得到的输出为“| < >”. 这是因为按 TeX 规定, 用美元号 \$ 括起来的部分处于数学模式(后面会详细介绍), 这 3 个字符可以在数学模式下打印.

此外, Plain $\text{T}_{\text{E}}\text{X}$ 中的特殊字符 “@” 在 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 中变成了普通字符, 可直接输入。

字符 “*” 是一个普通字符, 这是直接输入后排印显示的结果, 如果想要它显示在上下居中的位置如 “ $*$ ”, 可用数学模式输入成 $\$*\$$ 。

还有一些有用的符号可参见下表。

输出符号	§	¶	†	‡	©	£
输入序列	\S	\P	\dag	\ddag	\copyright	\pounds

$\text{T}_{\text{E}}\text{X}$ 、 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 与 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 等标志, 是通过输入 “ $\backslash\text{TeX}\{\}$ ”、“ $\backslash\text{LaTeX}\{\}$ ” 以及 “ $\backslash\text{LaTeXe}\{\}$ ” 得到的。

下面是使用特殊符号的例子: 为得到 “\$3.50”, 应该输入 “ $\backslash\$3.50$ ”; 为得到 “50%”, 应该输入 “50\%”。

2.2.2 空格与换行

我们知道, 西文单词是用空格隔开的, 句末的标点之后也应该留有空格。但应注意, $\text{T}_{\text{E}}\text{X}$ 源文件中的连续多个空格在编译排版时被看作是一个空格, 而且与许多“所见即所得”的排版软件 (例如 Word, WPS 等) 不同, 单个回车 (注意, 连续两个回车将被看成段落的结束, 参见 §2.4) 仅相当于一个空格, 与最终得到的版面没有关系。因此你可以放心换行, 不必担心最终结果, $\text{T}_{\text{E}}\text{X}$ 会自动使得左右两端对齐的。此外, 西文以后的空格标志着单词或句子的结束, 排版时会留有适当的间距, 汉字之后的空格在排版时是否起作用与所用 CJK 环境有关。原稿输入时换行最好选在句末标点以后。控制字后面的空格被认为是命令序列的结束标志, 排版输出时不会出现空格所对应的间隔, 如果在控制字后面紧接着放上 $\backslash_$, 即一个倒斜线和一个空格, 则它既表示命令序列的结束, 又会输出一个空格对应的间隔。控制符后面不需要任何结束标志, 因此在控制符后面的空格会输出相应的空白间隔。

$\text{T}_{\text{E}}\text{X}$ 编排版面时, 可能会在空格处换行。如果强制 $\text{T}_{\text{E}}\text{X}$ 不能在某个空格处换行, 则该空格需改用“不可打断的空格”, 在原稿中用符号 “~” 表示这种空格。这个符号在 CJK 环境中也起同样作用, 但在 CJK* 环境中 “~” 号被重新定义了, 不再起上述作用, 此时需用命令 $\backslash\text{ns}$ 代替 “~”, 见 §2.8 节。相连的不可打断的空格有几个算几个, 不会被当成一个空格。不可打断的空格既不可不用, 又不可滥用, $\text{T}_{\text{E}}\text{X}$ 作者认为, 能够正确使用 “~” 号是 $\text{T}_{\text{E}}\text{X}$ 高手的标志之一。

2.2.3 引号、连字号、破折号

在文稿中输入引号时, 左单引号用键盘上的倒引号 “ \backslash ”, 亦称重音号, 在键盘打字区左上角, 右单引号直接用键盘上的单引号 “ $'$ ”。

左双引号是连用两个左单引号, 右双引号是连用两个右单引号或直接用键盘上的双引号 “ $''$ ”, 它与单引号在同一个键上, 是上档键。

当单引号与双引号相邻时, 中间应插入一个小间隔, 命令 “ $\backslash,$ ” 产生的间隔就很合适。例如输入 “ $\backslash\backslash,$ $\backslash\text{single}'$ and $\backslash\text{double}'\backslash,$ ” 产生 “ $\text{'single}'$ and $\text{'double}'$ ”。

使用一个连字号“-”产生一个连字号“-”。

连用两个连字号“--”产生一个表示数字范围的符号“-”。

连用三个连字号“---”产生一个西文破折号“—”。

例如人名 Harish-Chandra 中间是连字符 (hyphen), 而 pages 2-10 中间则是表示数字范围的短线。

为了得到数学中的负号或减号“-”, 需用数学模式输入成 $-$$ 。

利用中文输入法, 可以输入各种中文标点符号。CJK 将中文标点符号当作普通汉字处理, 不具有特殊的控制符号功能。

2.2.4 连体字

在印刷的书籍中, 有些特殊的字母组合, 它们不是一个一个地印出来, 而是作为一个连体符号排印。T_EX 将字母组合 ff, fi, fl, ffi 以及 ffl 作为连体字显示为 ff, fi, fl, ffi, ffl, 而不是 ff, fi, fl, ffi, ffl。通常情况下这样处理是很好的, 但有时不作为连体字更好, 例如单词 shelfful 中的两个 f 按连体字排版时显示为 shelffful, 就不大美观了。若要将连体字强制分开, 可插入命令“\”, 上述单词应输入成 shelf\ful。或者将其中一个 f 用花括号括起来, 输入成 shelf{f}ul。

有些字母相邻时, 相互距离会拉近, 例如 AV 和 Te 等。插入命令 \ 可取消这种距离的缩短, 例如输入 A\V 和 T\ e 得到 AV 和 Te。或者是将其中一个字母用花括号括起来, 输入成 A{V}、T{e} 或 {A}V、{T}e。

2.2.5 句号后的空白

句号圆点有很多用处, 其中的两个用处: 一是用在句末表示句子的结束, 二是表示缩写。这两种圆点后面的空格排版出来的空白长度应有所不同, 表示句子结束的空白应更长一些。T_EX 确实是在句号后面插入了额外的空白间隔, 但是对于以小写字母结束的缩写, T_EX 系统并不能自动认出它是缩写, 仍当成句子结束处理。为了“通知”T_EX 系统圆点表示缩写, 需将圆点后的空格改成“_”, 即在空格前面加上倒斜线, 这时间隔就是正常的了, 不会插入额外空白。还可以用“~”号取代普通空格, 它除了具有_的作用之外, 还禁止在该处分行。所以句子“Prof. Jones read the Phys. Rev.,” 应输入成“Prof.~Jones_read_the_Phys._Rev.,”, 其中第一个缩写后的空格改用了“不可打断的空格”, 因为称呼和姓名不应分在两行, 第二个缩写后的空格使用了“_”, 既表示圆点不是句号, 又表示必要时该处可以分行。注意在 CJK* 环境中需用命令 \nbs 产生不可打断的空格。

T_EX 将紧接在大写字母后的圆点看作是一个缩写而不认为是句号。但有时大写字母后的圆点确实是句号, 这时就需要在圆点前加上“\@”, 以通知 T_EX 按句号处理, 得到附加的空白。例如“We work in ECNU.” 应输入成“We work in ECNU\@.”。

使用命令

```
\frenchspacing
```

句点后面就不再有额外留空. 取消这个命令的命令是 `\nonfrenchspacing`.

2.2.6 非英文字母及重音符号

在非英文的西文文字中有一些特殊的字符, 可按下表输入:

œ {\\oe}	Œ {\\OE}	æ {\\ae}	Æ {\\AE}	å {\\aa}
Å {\\AA}	ø {\\o}	Ø {\\O}	ı {\\l}	Ł {\\L}
ß {\\ss}	SS {\\SS}	ı !`	ı ?`	

表中的命令都用花括号括起来了, 这样在一个单词中输入特殊字符时, 可以与前后字符紧靠在一起, 而不必用空格来标志命令的结束, 从而在文稿上不会将一个单词错看成是两个单词. 例如将 `schließen` 输入成 `schlie{\ss}en` 比 `schlie\ss en` 更好看一些.

有些语言的字母带有各种发音记号(重音符号), 以字母 `o` 为例, 如下输入:

ò \\o	ó \\o	ô \\o	ö \\o	õ \\o
ō \\=o	ô \\o	ö \\u{o}	ö \\v{o}	ő \\H{o}
ô \\r{o}	ôo \\t{oo}	o \\b{o}	o \\c{o}	o \\d{o}

从表中可以看出, 对于由非字母组成的重音命令, 可以不使用花括号, 但由字母组成的重音命令就必须用花括号.

如果要在字母 `i` 和 `j` 的顶上加记号, 应先去掉原有的点, 可通过命令 `\i` 和 `\j` 得到 `i` 和 `j`, 所以 `ï` 与 `ĵ` 是通过 `\v{\i}` 和 `\H{\j}` 得到的.

如果需要希腊字母或各种数学符号, 请参阅第四章.

§ 2.3 分组与环境

不同的命令有不同的作用范围, 有些命令只对其后的一个字符起作用, 有些命令对其后的所有文本起作用. 为了扩展或限定命令的作用范围, 常常需要用一对花括号将一些文本括起来, 称为一个“分组”或一个“集团”. 对于那些只对其后的一个字符起作用的命令, 如果其后是一个分组, 则它就对整个分组起作用. 实际上这样的命令就是带参数的命令, 只是当参数为单个字符时, 可不必用花括号把参数括起来. 例如 `This is \textbf{bold face} style` 输出为 `This is bold face style`, 可见命令 `\textbf` 对其后的整个分组起作用, 对其他位置的字符不起作用. 对于那些对其后的所有文本起作用的命令, 即不带参数的命令(有时称为“声明”), 为了限定作用范围, 可将它们放在一个分组的内部, 它的功能到该分组的结束符“`}`”处就不起作用了. 例如 `{这是\CJKfamily{hei}中文黑体}` 的输出效果是: 这是中文黑体, 可见命令“`\CJKfamily{hei}`”对它前面的字符以及分组外部的字符是不起作用的.

分组时最常见的错误有两个, 一是记混了上述两种命令的作用范围, 二是漏掉了分组结束的花括号.

在 L^AT_EX 中, 为了对某些文本进行特定格式的排版, 需要把它们放到相应的环境中, 语法是:

```
\begin{环境名}
  文本
\end{环境名}
```

其中开始和结束的环境名必须相同. 在环境内的文本中可以含有其他命令和声明, 通常它们的作用范围终止于环境的结束.

§ 2.4 分段

在源文件中的一个空行 (连续按两次回车产生一个空行) 相当于一个排版分段命令, T_EX 排版输出时, 空行的上方和下方文稿会被排成两个段落, 这两个段落之间并不出现空行. 源文件中连续多个空行与一个空行的作用相同.

如果不愿意在源文件中用空行来控制分段, 则可用分段命令: 在需要分段的地方插入命令 “\par”, 该命令的前后文稿就会被排成两段.

可以控制两段之间的间隔与两行之间的间隔相同或不同, 也可以控制段落首行缩进或不缩进. 习惯上中文文章段落间隔与行间隔相同, 并且首行缩进, 而西文文章通常两段之间有较大间隔且首行不缩进. 与段落有关的一些间隔设置见 § 2.7 节.

§ 2.5 分行和分页

当对 T_EX 排版输出的结果不甚满意时, 可以适当进行调整.

2.5.1 分行

强制分行的排版命令是

```
\\      或      \\*      或      \newline
```

命令后面的文字会被新起一行从头排起. 这与分段不同, 因为分行的文字仍属于同一段落, 不受段落间距

及首行缩进的影响. 上一句突然中断换行就是因为插入了命令 \\. 带星号的 * 除了强制分行外, 还禁止在分行处分页.

命令 “\\” 还可以带有参数, 形如 “\\[长度]” 或 “*[长度]”, 用来增加或减少行间隔. 例如 \\[5cm] 使当前行与新行之间增加 5 cm 的间隔, 而 \\[-1mm] 使当前行与新行之间的间隔 (文稿中在此处放置了命令 \\[-1mm]) 减少了 1 mm. 我想你已经能看出命令 \\[-1mm] 的效果了.

当分页位置正好在当前行和新行之间时, “\\[长度]” 与 “*[长度]” 就有区别了, 前者相当于变成了一个分页命令, 使当前行与新行放在两页里, 指定的间隔被

“吃掉”了不再起作用. 而后者会在当前行之前分页, 使得当前行和新行同时出现在新的一页里, 并仍保持指定的间隔. 当指定的间隔超出一页的高度时, 当前行出现在新页的顶部, 新行出现在下一页的顶部.

另一个分行命令是“建议分行”的命令

`\linebreak[数字]`

其中数字是0到4之间的一个整数. 数字越大, 建议的力度也越大, 使用数字4时已不是建议分行而是强制分行, 此时命令可以简写为`\linebreak`, 使用其他数字时, 排版系统根据内部设定的不同阈值, 决定是否采纳分行建议. 与`\`、`\newline`有所不同, `\linebreak`形式的分行命令的输出结果是使当前行撑满整行, 行内单词或汉字之间的间距可能变得很大. 读者一定会发现上一行排得特别空松, 这就是插入了命令`\linebreak`后产生的效果.

与建议分行命令相反的是“建议不分行”命令

`\nolinebreak[数字]`

同样是数字越大, 建议的力度也越大, 使用数字4(或不带参数)时已不是建议, 而是强制不允许在当前位置分行了.

有时需要将一些文本整体保持在同一行中, 不允许在中间任何地方分行, 可使用命令

`\mbox{文本}`

注意不要将太多的文本保持在同一行中, 以免出现超长的行使得输出版面难看.

如果排版后某行末尾的一个外文单词使得该行超长, 可在该词内部插入一个或几个`\-`, 这是建议断词的位置, 断词后一个单词分在了两行, 上一行的末尾会自动插入一个连字符.

2.5.2 分页

通常情况下 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 会自动分页, 无需人工干预. 必要时可强制在指定位置分页. 类似于分行的`\newline`命令, 强制分页的命令是

`\newpage`

如果该命令出现在两个段落之间, 则这前后两段就被排版到两页, 并且当上一段所在页面不满时, 该页面下部会出现空白. 如果该命令出现在一段之内, 则在排版输出时会在命令所在位置把一段强制变成两段, 然后在这两段之间强制分页.

建议分页或建议不分页的命令是

```
\pagebreak[数字]  
\nopagebreak[数字]
```

介于 0 与 4 之间的参数表达了建议的力度, 当参数数字是 4 时, 可以省略, 此时建议性的命令实际变成了强制性的命令。

用 `\pagebreak` 分页与用 `\newpage` 分页的效果是不同的。当在两段之间出现 `\pagebreak` 命令时, 虽然也是在这两段之间进行分页, 但当上一段所在页面不满时, 会自动增加段落间距以撑满一页。如果该命令出现在一段之内, 则在排版输出时会在当前行完成后再分页。

命令 `\nopagebreak` 出现在两段之间时禁止在该处分页, 出现在一段之内时, 禁止在当前行完成后出现分页。

有时稍稍增加当前页的高度就可能避免一个难看的分页, 命令

```
\enlargethispage{尺寸}  
\enlargethispage*{尺寸}
```

可使当前页 (仅仅是当前页) 增加一点高度。不带 * 号的命令指定的尺寸是当前页可以增加的最大高度, 实际增加值可能小于指定值。* 形式命令使当前页严格增加指定的高度, 因此可能会使行间隔和段落间隔发生少许变化。

§ 2.6 水平间距、竖直间距

2.6.1 水平间距

命令 `\,` (即倒斜线后紧跟一个逗号) 产生一个很小的水平间距, 大约是字母 M 宽度的 1/6。如 99 999 中间的空格就是 `\,` 的效果。

产生指定长度的空白可以使用下述命令

```
\hspace{长度}  
\hspace*{长度}
```

两种格式在通常情况下没有区别, 都是在文本中间插入水平空白间隔, 但是当排版结果恰好使得开始插入的位置是在新行的开始, 则不带 * 号的命令插入的空白会被吃掉 (相当于没写这个命令), 而带有 * 号的命令在任何情况下都会插入指定长度的空白间隔。命令中的长度可以是负值, 这会使命令后面的文本向回退, 当这个负值的绝对值较大时, 会使命令后面的文本退到前面文本的左面产生重叠现象。

下面是两个与当前所用字样 (font) 有关的插入固定长度的命令

```
\quad      \qquad
```

命令 `\quad` 插入相当于当前字样尺寸的空白, 例如当前使用 10 pt 字样时, `\quad` 相当于 `\hspace{10pt}`, 而 `\qqquad` 是 `\quad` 的两倍. 例如

“始 中 尾” 里面的空格就是 `\quad` 与 `\qqquad` 的效果.

有时为了撑满一行, 需要插入未知长度的空白, 这时可以使用特殊的弹性长度 `\fill`, 它具有无限伸缩的能力, 最短时长度为零, 最长时需要多长有多长.

命令 `\hspace{\fill}`, 简写为

```
\hfill
```

这个命令会根据排版的结果, 在命令两边的文本之间插入需要的空白, 以撑满一行. 当这个命令位于多行段落之内时, 通常是不起作用的, 因为每行都是满的, 但当它位于单独的一行或接近段落的末尾时, 就容易看到它的效果了, 这时弹性长度起到了弹簧的作用, 使所在行的文本撑满一行.

当 `\hfill` 位于一行开头或者末尾时, 空白会被吃掉, 就好像一根弹簧, 若一端没有支撑物, 另一端就不能顶开其他物体. 此时可在空白的一端放上一个没有宽度的文本如空盒子 `\mbox{}` 作“支撑物”, 或者换用带 * 号的命令 `\hspace*{\fill}`.

利用 `\hfill` 容易使单行文本左对齐、右对齐或居中对齐. 例如输入

```
这是靠左对齐的一小段文本\hspace*{\fill}\\
\hspace*{\fill}这是居中对齐的一小段文本\hspace*{\fill}\\
\hspace*{\fill}这是靠右对齐的一小段文本
```

得到

```
这是靠左对齐的一小段文本
      这是居中对齐的一小段文本
                        这是靠右对齐的一小段文本
```

对单行或多行文本作对齐处理, 有现成的命令和环境, 详见第 32 页和第 107 页. 上例介绍的对齐方法大多用于表格的单元格中, 如果要使某一单元格不按着表格环境预定的列格式对齐文本, 上述对齐方法就有用武之地了.

在 TeX 中有一个占位命令

```
\hphantom{文本}
```

它精确地占据了文本的宽度但不显示文本的内容. 当需要的水平间距恰是某段文本的宽度时, 使用这个命令比使用 `\hspace` 要方便得多, 因为它无须测量这段宽度的具体数值. 该命令占据的高度为零, 所以不会影响行距. 类似的, `\vphantom{文本}` 只占据文本的高度, 而不占据宽度. `` 则占据文本的区域: 既占据高度, 又占据宽度, 但不显示文本.

2.6.2 导引线

命令

```
\dotfill      \hrulefill
```

具有类似于\hfill的功能,但它们不是产生单纯的空白,而是分别用点线和实线填充空白.例如输入

```
左端文本\hrulefill 中间文本\dotfill 右端文本
```

得到

```
左端文本_____中间文本.....右端文本
```

在目录标题和对应页码之间常常是用点线连接起来,这种线起了指引作用,所以把填充空白的点线或实线称作导引线.

若用点线填充指定长度的空白,可参见下面一行的输入和输出:

```
\makebox[3cm]{\dotfill}      ⇒      .....
```

2.6.3 竖直间距

使用命令

```
\vspace{长度}  
\vspace*{长度}
```

可以在两行之间插入竖直间隔.需要注意的是这两个命令不起分行的作用,即不会自动将命令前后的文本分成两行,只有在当前行完成后才在该行下面插入竖直空白间隔.这一行与上一行的行距增大就是由于上一行中间插入了命令\vspace{1mm}.

与水平间隔类似,当插入的空白恰在新一页的开始时,不带*号的命令插入的间隔被吃掉,而带*号的命令在任何情况下都会插入指定的空白间距.插入竖直间隔的命令大多用于两个段落之间.命令中的长度也可以是负值,使下一段的位置向上提升.

与\hfill对应的是

```
\vfill
```

即\vspace{fill},当一页不满时,它会插入足够的空白以撑满一页.

有三个内部定义好的具有弹性高度的命令可以插入竖直空白,它们按正常值排列从小到大是

```
\smallskip    \medskip    \bigskip
```

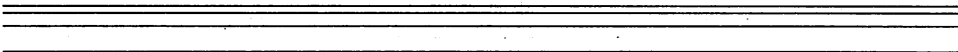
它们的定义是

```
\vspace{\smallskipamount}  
\vspace{\medskipamount}  
\vspace{\bigskipamount}
```

其中的三个长度参数由具体的文档版式所给定, 用户也可重新定义, 例如可如下将“小间隔”重新定义为基本长度 2 mm, 最多 3 mm, 最少 1.4 mm:

```
\setlength{\smallskipamount}{2mm plus 1mm minus 0.6mm}
```

为了让读者有个直观印象, 下面画了 4 条线, 它们之间的距离相当于本书的版式所规定的 3 种间隔:



§ 2.7 与段落有关的距离

2.7.1 首行缩进

段落首行缩进的距离由长度 `\parindent` 决定, 用户可如下改变这个值:

```
\setlength{\parindent}{长度}
```

当上述长度不是零时, 段落的首行会自动缩进由长度指定的距离. 对于中文文章, 命令

```
\setlength{\parindent}{2em}
```

可使每段首行缩进两个汉字的距离. 如果使用 4.5.2 版以后的 CJK, 则无需自己设置 `\parindent` 的值, 只须在 CJK 环境中写上

```
\CJKindent
```

就会使其后段落首行缩进两个汉字的距离. 需要注意的是上述两种首行缩进的距离等于当前字体两个汉字的宽度, 若改变了字体尺寸, 需重新再写一次上述命令, 才会使后继段落首行缩进两个新字体汉字的宽度.

若想使某段首行不缩进, 可在该段开始放上命令 `\noindent`.

需要注意的是每一节的第一段的首行并不会缩进, 为了使第一段能像其他段一样会首行缩进, 可在第一段开始放上命令

```
\hspace*{\parindent}
```

或更简单地在源文件的导言区放上命令

```
\usepackage{indentfirst}
```


2.7.2 段落间距

两段之间的距离等于行间隔 (也称为行隙) (`\lineskip`) 加上长度 `\parskip` 的值, 如果重设这个值, 通常应设置成弹性长度并使用 `ex` 作单位, 以便它随着字体尺寸的改变而变化。

中文书籍段落之间的距离通常与行间隔相同, 为达到这种效果, 可在导言区放上命令

```
\setlength{\parskip}{0pt}
```

2.7.3 伸展行距

行距 (`\baselineskip`) 是相邻两行基线之间的距离。所谓基线可粗略地看成是一行文字底部的直线, 例如一行字母 `abxy` 中的 `abx` 座落在基线上, 但字母 `y` 的下伸部分落在了基线下方。

当选定了一种字体尺寸, 行距就被自动确定下来了。如果要伸展行距, 可通过重新设置伸展因子达到目的, 命令格式为:

```
\renewcommand{\baselinestretch}{伸展因子}
```

其中伸展因子的值是一个十进制小数, 新的行距是原有基本行距乘以这个因子后的值。

上述命令写在导言区时直接对整个文本起作用, 但若放在正文中 (即在命令 `\begin{document}` 之后), 则不会立即起作用, 只有当字体尺寸改变时 (遇到一条选择字体尺寸的命令), 伸展因子才起作用, 而且是从字体尺寸命令所在的段落开始起作用, 不影响前面的段落。

如果只想改变行距而不改变字体尺寸, 那么只要紧接在伸展行距命令之后再写上原有的字体尺寸命令即可。当文稿中未出现字体尺寸命令时, 默认是正常大小, 即相当于使用了命令 `\normalsize`。此时若把以后的行距改为“双行”行距但不改变字体尺寸, 可同时使用以下两条命令

```
\renewcommand{\baselinestretch}{1.6} \normalsize
```

此例伸展因子用 1.6, 排版后的视觉效果较好, 若用 2.0, 则显得过宽。

§ 2.8 CJK 小结

前面零星地介绍了一些有关 CJK 的知识, 本节再补充一些材料, 作个小结, 基本只涉及中文。有些内容目前可能觉得难以理解, 稍有印象即可, 等到需要时再回过头来细看。

首先我们应该在导言区加进读入宏包的命令:

```
\usepackage{CJK}
```

这样就定义了两个新的环境:

```
\begin{CJK}[字模编码]{编码}{族}
.....
\end{CJK}
```

以及

```
\begin{CJK*}[字模编码]{编码}{族}
.....
\end{CJK*}
```

这里的编码与汉字有关的主要有以下几种:

GB 国标码 GB 2312-1980;
 GBK GBK 大字符集 GB 13000;
 Bg5 繁体汉字大五码;
 Gbt 繁体汉字扩展码 GB/T 12345-1990.

族则取决于你自己的定义, 我们已经替你预先定义了 GBK 编码下的 song (宋), fs (仿宋), kai (楷), hei (黑), li (隶), you (幼圆) 六个族, 这是因为简体中文 Windows 一般提供这 6 种字体, 因此建议你总是使用 GBK 编码.

可选项字模编码可以忽略.

环境 CJK* 自动忽略汉字后面的空格以及换行, 只考虑像 “_” 那样受保护的空白; 在 CJK 环境中, 汉字后面普通的或受保护的空白都不被忽略. 一般来说, 使用 CJK* 环境时, 不要忘记在中西文交界处写上受保护的空白或符号~, 使用 CJK 环境时, 不要在中文之间输入空格, 至于使用哪种环境, 取决于个人的偏好. 对汉字后面空格的处理, 还可以在环境内部通过下述命令随时切换.

```
\CJKspace            \CJKnospace
```

前者把 CJK* 转换成 CJK, 后者则与之相反.

任何一种 CJK 环境的内部可以再嵌套任何 CJK 环境, 但是这会耗费 T_EX 处理时的堆栈空间, 而且实际上也没有必要, 因为下面的几条命令可以在两种 CJK 环境的内部切换汉字的编码以及族. 如

```
\CJKenc{编码}
\CJKfamily{族}
```

能改变以后出现的汉字的编码及族, 例如 \CJKenc{GB}, \CJKfamily{fs} 等. 为了使用方便, 你可以自己定义新的命令如:

```
\newcommand{\SONG}{\CJKfamily{song}}
\newcommand{\HEI}{\CJKfamily{hei}}
\newcommand{\KAI}{\CJKfamily{kai}}
\newcommand{\FS}{\CJKfamily{fs}}
\newcommand{\LI}{\CJKfamily{li}}
```

以后只要发出命令 `\KAI` 就能把字体转换成楷体。

遇到西文或数学公式夹在汉字中间时,在交界处输入小间隔“\,”,就可得到较为满意的输出。在 CJK* 环境里,如果在交界处紧接着输入,不留空格,例如输入“变量 x 的值”,排版结果得到“变量 x 的值”;如果在交界处留空,例如输入“变量 x 的值”,排版结果得到“变量 x 的值”。前一种情形显得太拥挤,后一种情形更糟,前紧后松。为了解决这个问题,可使用以下命令:

```
\CJKtilde
```

这个命令把不可分割空格“~”重新定义为

```
\def~{\hspace{0.25em plus 0.125em minus 0.08em}}
```

这样就能够得到合理的间距,例如输入“变量 x ~的值”可以得到输出“变量 x 的值”。“~”比“\,”产生的间隔略大一些。如果你想使“~”恢复原来的定义,可以使用以下命令

```
\standardtilde..
```

如果你偶尔需要使用“~”原来的效果,但又不想使“~”恢复原来的定义,那么可以用以下命令代替原有的“~”:

```
\nbs
```

这里的 `\nbs` 可以看成是 L^AT_EX 命令 `\nobreakspace` 的缩写。

在 CJK 环境中,如果两个汉字之间夹有多个英文单词,可以直接在中英文之间输入一个空格,排版结果是比较好看的;如果汉字之间只有单个的英文单词,则中英文之间插入小间隔\,较好,当要求不是很高时,插入空格也是可以的,这样输入起来很方便。

有了这些命令后,我们看一个使用 CJK* 环境的例子(参看 2-8-1.tex):

```
% 2-8-1.tex
\documentclass[11pt]{article}
\usepackage{CJK}
\newcommand{\SONG}{\CJKfamily{song}}
\newcommand{\HEI}{\CJKfamily{hei}}
\newcommand{\KAI}{\CJKfamily{kai}}
```

```

\newcommand{\FS}{\CJKfamily{fs}}
\begin{document}
\begin{CJK*}{GBK}{song}
\CJKtilde
这是宋体字, {\HEI 这是黑体字}

{\KAI 这是楷体字, \textbf{这是粗楷体字}}

{\FS 这是仿宋体字, \textbf{这是粗仿宋体字}}
\bigskip

\textbf{Fermat~定理:} 对~$n\ge3$,
$$
\left\{(x,y,z)\in\mathbf{Q}^3\mid x^n + y^n =
z^n, xyz\ne0\right\} = \emptyset.
$$
\end{CJK*}
\end{document}

```

请注意 `\textbf` 产生汉字粗体的方法是通过对这个汉字重复打印而实现的, 即在水平方向上作轻微移动, 这被称为“穷人的粗体” (poor-man's boldface). 命令

```
\CJKboldshift
```

规定了生成粗体的移动距离, 其默认值是 0.015 em, 如果觉得不满意, 可以重新定义, 例如

```
\renewcommand{\CJKboldshift}{0.02em}
```

在目前的 CJK 版本中, `\textit` 和 `\textsl` 对汉字的效果是一样的, 都使它变成斜体, 其余字体变化命令均对汉字没有影响. 还要注意, 对汉字来说, 粗体命令 `\textbf` 是一种字形命令, 用于加粗当前的文字. 而族 `hei` 则是与族 `song`、`kai`、`fs` 等并列的一种字体 (黑体).

下面再介绍一些有用的 CJK 命令.

```
\CJKglue
```

这个命令的原始定义是:

```
\newcommand{\CJKglue}{\hskip 0pt plus 0.08\baselineskip}
```

它于必要时在汉字之间插入一个附加的空隙, 以解决行的超长. 如果你遇到行超长无法解决时, 可以修改这个命令, 增大它的值, 以加强其调节功能.

```
\CJKkern
```

此命令阻止在这两个汉字间换行, 它取消了命令前面的 `\CJKglue` 的作用.

此外, `verbatim` 环境在 CJK 内仍然有效.

进一步的用法请参看 `texmf\doc\latex\Cjk` 目录中的 `commands.doc` 以及 `CJK.doc`, 这是两个文本文件, 不需使用 Word.

§ 2.9 天元文稿转换成CJK文稿

本节介绍如何将天元文稿转换成 CJK 文稿, 转换后的效果虽然不能保证完全相同, 但基本上是相似的. 转换前的源文件扩展名大多为 `.ty`, 转换后的扩展名应使用 `.tex`.

2.9.1 增删一些语句

删除 (或用 % 注释掉) 天元文稿开头部分的如下两行:

```
\def\ChineseScale{1000}
\input tyinput
```

其中第一行中的数字也可能是 1095, 1200, 1440 等.

在 `\begin{document}` 之下添加语句 `\begin{CJK}{GBK}{song}`.

在 `\end{document}` 之上添加语句 `\newpage` 和 `\end{CJK}`.

2.9.2 转换字体命令

按着表 2.1 列出的对应关系将天元文稿中的字体命令换成对应的 CJK 字体命令.

表 2.1 天元字体命令

天元字体命令	CJK 字体命令
<code>\宋</code>	<code>\CJKfamily{song}</code>
<code>\黑</code>	<code>\CJKfamily{hei}</code>
<code>\楷</code>	<code>\CJKfamily{kai}</code>
<code>\仿</code>	<code>\CJKfamily{fs}</code>
<code>\隶</code>	<code>\CJKfamily{li}</code>
<code>\圆</code>	<code>\CJKfamily{you}</code>

天元软件定义了近 20 个字体命令, 除了表 2.1 列出的几种以外, 还有 `\题`、`\行`、`\姚`、`\美`、`\准`、`\魏`、`\变`、`\琥`、`\综` 等多个字体命令, 必须自行安装相应的字体, 并改成相应的 `\CJKfamily{...}` 命令, 才能在 CJK 环境使用这些字体.

2.9.3 转换字体尺寸命令

按着表 2.2 列出的对应关系将天元文稿中的字体尺寸命令换成对应的 L^AT_EX 字体尺寸命令.

表 2.2 天元字体尺寸命令

天元命令	L ^A T _E X 命令
\五 或 \半	\tiny
\六	\tiny 或 \scriptsize
\七	\scriptsize
\八 或 \小	\footnotesize
\九	\small
\标	\normalsize
\中	\large
\大	\Large
\特	\LARGE
\双	\huge
\巨	\Huge
\叁	\newcommand{\SANSHI}{\fontsize{30}{36}\selectfont} \SANSHI
\肆	\newcommand{\SISHI}{\fontsize{40}{48}\selectfont} \SISHI
\伍	\newcommand{\WUSHI}{\fontsize{50}{60}\selectfont} \WUSHI
\陆	\newcommand{\LIUSHI}{\fontsize{60}{72}\selectfont} \LIUSHI

如果需要按表 2.2 定义新的字体尺寸命令, 只需定义一次.

2.9.4 转换字形命令

在天元软件中定义了 \扁、\瘦、\阔、\正 4 种汉字字形命令, 在 L^AT_EX 中没有类似的命令. 为了实现类似的功能, 可用第十二章介绍的图形缩放命令, 见表 2.3.

表 2.3 天元字形命令

天元命令	L ^A T _E X 命令
\扁	\scalebox{1.0}[0.8]{汉字}
\瘦	\scalebox{0.8}[1.0]{汉字}
\阔	\scalebox{1.2}[1.0]{汉字}

表 2.3 中没有提供对应于 \正的命令, 该命令用于取消其他 3 个天元字形命令的作用. 由于图形缩放命令只对其后放在花括号中的文字起作用, 因此不再需要对应于 \正的命令.

第三章 文字模式

L^AT_EX 把排版情况分成 3 种模式: 段落模式、左到右模式和数学模式. 段落模式就是把文稿分行、分段和分页, 排成需要的版面; 左到右模式是把输入的字符排成从左到右的一行, 无论长短都不分行; 数学模式用于排版数学公式. 我们把前两种模式统称为文字模式放在本章中一起介绍.

为了版面美观和其他需要, 一篇文稿中大多会使用几种不同的字体. 所谓字体就是文字的风格样式. 具有特定大小和外观的一组字母、数字和符号的集合构成一种特定字体的字符集. 在 L^AT_EX 中用于文字模式的默认字体是直立的罗马字体. 在 CJK 中使用 GBK 编码时, 默认的中文字体是宋体, 即当 CJK 环境的族参数为空时, 相当于该参数是 `song`.

§ 3.1 西文字体

西文字体的基本尺寸是 10 pt、11 pt (严格说是 10.95 pt) 和 12 pt, 从下面 3 行可以看出 3 种尺寸的差别:

This is the 10 pt font. ABXYabxy1234 这是 10 pt 大小
This is the 11 pt font. ABXYabxy1234 这是 11 pt 大小
This is the 12 pt font. ABXYabxy1234 这是 12 pt 大小

3.1.1 字体属性

字体尺寸大小仅是字体的一种属性, 在新字体选择方案 (NFSS) 中, 每种字体有 5 种属性: 编码、族、系列、形状和尺寸. 普通用户一般不会涉及字体的编码.

族 (family) 指的是概观样式. L^AT_EX 提供了 3 种声明用于选择不同的族:

`\rmfamily` 切换到罗马 (roman) 字体
`\sffamily` 切换到无衬线 (sans serif) 字体
`\ttfamily` 切换到打字机 (typewriter) 字体

形状 (shape) 指的是倾斜和高矮, 在 L^AT_EX 中提供了 4 种声明用于选择不同的形状:

`\upshape` 切换到直立 (upstanding) 字体
`\itshape` 切换到意大利 (*italic*) 斜体
`\slshape` 切换成称为 *slanted* 的斜体

`\scshape` 切换成小体大写 (SMALL CAPS) 字体

系列 (series) 指的是字体的宽度和权重 (黑度), 权重反映了笔画的粗细. 可用的声明有:

`\mdseries` 切换到中等 (medium) 权重

`\bfseries` 切换到粗体 (bold face)

上述一些命令称为声明, 就是说这些声明在遇到新的声明之前一直起作用. 为了限定其作用范围, 应把它放在一组内, 即用花括号把声明和受影响的文本括起来, 或者对于很长的文本使用相应的环境

```
\begin{字体属性}
    使用新属性的文本
\end{字体属性}
```

其中的字体属性可以是上面任何一种属性声明, 但要去掉前缀 “\”.

还有一个非常重要的声明 `\normalfont`, 它把除了字体尺寸以外的所有属性重设成默认值, 即中等权重的直立的罗马字体.

对应于上面的字体声明, 都有相应的字体命令, 这些命令只对命令参数中的文本起作用, 当改变一小段文本或一个单词的字体属性时, 建议使用字体命令而不使用字体声明. 改变属性的命令有:

```
族:      \textrm{文本} \textsf{文本} \texttt{文本}
形状:    \textup{文本} \textit{文本} \textsl{文本} \textsc{文本}
系列:    \textmd{文本} \textbf{文本}
默认值:  \textnormal{文本}
强调:    \emph{文本}
```

这些命令中的文本不能位于两个段落中. 上述强调命令将参数中的字体改变成强调字体: 如果当前字体是直立字体, 强调字体就是 *italic* 斜体, 而如果当前字体是斜体 (*italic* 或 *slanted*), 则强调字体就是直立字体. 与强调命令对应的强调声明是 `\em`.

当由斜体切换成直体时, 斜体字符与直体字符之间的间距会显得比通常的字符间距小, 因此应根据不同的字符而插入一些额外的间距 (*dh* 之间应比 *bh* 之间插入更多一点间距). \TeX 用命令 `\/` 加入这个间距, 称为倾斜校正. 强调命令 `\emph` 会自动插入倾斜校正, 而用强调声明 `\em` 时必须人工加入倾斜校正. 如果使用强调命令时不要倾斜校正, 应在倾斜字符后面加上命令 `\nocorr`.

slanted 字体与 *italic* 字体都是倾斜字体, 但两者字形稍有不同, 倾斜度也不同, 前者向右倾斜了高度的 $1/6$, 后者向右倾斜了 $1/4$.

3.1.2 选择字体尺寸

当在文档类选项中指定了基准尺寸即基本字体的大小后, 可用下面的声明 (也称为字体尺寸命令) 来改变字体的大小. 表中命令后面的文本是在基准尺寸为 10 pt 的情况下, 相应声明对应的字体大小的示例:

<code>\tiny</code>	5pt, smallest	<code>\scriptsize</code>	7pt, very small
<code>\footnotesize</code>	8pt, smaller	<code>\small</code>	9pt, small
<code>\normalsize</code>	10pt, normal	<code>\large</code>	12pt, large
<code>\Large</code>	14.4pt, larger	<code>\LARGE</code>	17.28pt
<code>\huge</code>	20.74pt	<code>\Huge</code>	24.88pt

基准尺寸不同时各字体尺寸命令对应的大小(单位是pt, 已取整):

字体尺寸命令	基准尺寸为 10pt	基准尺寸为 11pt	基准尺寸为 12pt
<code>\tiny</code>	5	6	6
<code>\scriptsize</code>	7	8	8
<code>\footnotesize</code>	8	9	10
<code>\small</code>	9	10	11
<code>\normalsize</code>	10	11	12
<code>\large</code>	12	12	14
<code>\Large</code>	14	14	17
<code>\LARGE</code>	17	17	21
<code>\huge</code>	21	21	25
<code>\Huge</code>	25	25	25

如果要限制字体尺寸声明的作用范围, 应把它放到一个分组中, 或放在任何一个环境内部. 上述字体尺寸声明对中西文都起作用.

在 \LaTeX 中上述声明仅改变字体尺寸而不改变其他属性, 但在老版本的 $\text{\LaTeX}2.09$ 中, 上述字体尺寸声明在改变字体大小的同时, 还把其他属性重设成默认值, 相当于使用了声明 `\normalfont`.

每种字体实际上都有两个尺寸, 第一个尺寸表示字体大小, 有时简单地称为字号, 通常提到字体尺寸时, 大多是指字号; 第二个尺寸表示行距, 也就是两行基线之间的距离, 称为字体的自然行距, 对于 \LaTeX 实际安装的字体, 第二个尺寸总是大于第一个尺寸. 通常情况下排版时控制行距的命令 `\baselineskip` 的值就取自这个自然行距.

如果对自然行距不满意, 可以随时改变 `\baselineskip` 的值. 例如, 当前自然行距是 12pt, 命令

```
\setlength{\baselineskip}{20pt}
```

就把行距改成了 20pt, 看起来就是双倍行距的效果.

需要注意的是, `\baselineskip` 的值对整个段落起作用, 而且一个段落中只使用一个 `\baselineskip` 的值, 如果一个段落中这个命令的值改变了多次, 只有最后一次的值对本段起作用.

由于字体尺寸属性含有两个值, 所以每当使用了字体尺寸声明, 就相当于改变了两个值, 除了字体大小改变外, `\baselineskip` 的值也被重置为该字体尺寸对应的自然行距.

如果在一段落中使用了不同尺寸的字体, 并且没有使用花括号限制字体声明的作用范围, 那么当最后使用的是最大的字体时, 可以明显看出整段都是大的行距. 如果把最小的字体放在最后, 则意味着整段都用最小的行距, 此时不必担心前面相邻两行的大字体出现重叠现象, 这是因为 TeX 除了使用 `\baselineskip` 外, 还使用两个值 `\lineskiplimit` 和 `\lineskip`. 当相邻两行太靠近时——上一行的底部与下一行的顶部之间的距离 (称为行间隔) 小于 `\lineskiplimit` 时, 就强制这个距离等于 `\lineskip` 的值, 从而避免两行重叠.

调整行距时, 不推荐直接修改 `\baselineskip` 的值, 最好是使用伸展因子 `\baselinestretch`, 其正常值为 1. 实际上真正的行距是

$$\backslash\text{baselineskip} \times \backslash\text{baselinestretch}$$

用户可以用以下的命令

$$\backslash\text{renewcommand}\{\backslash\text{baselinestretch}\}\{\text{伸展因子}\}$$

随时改变这个因子的值, 但它的新值只有在又一次遇到字体尺寸声明时才起作用.

§ 3.2 中文字体

利用 CJK 宏包, 可以使用多种字体, 本书附盘已预先定义了 GBK 编码下的 song (宋), fs (仿宋), kai (楷), hei (黑), li (隶), you (幼圆) 六个字体族, 用户也可自行添加新的族.

在 CJK 环境中, 目前只提供了一种中文斜体, 两类斜体 (*italic* 斜体与 *slanted* 斜体) 命令对汉字的作用没有区别. 所有的字体尺寸命令、粗体命令以及强调命令对中文和西文同样起作用.

中文书籍基本字号是五号字, 大小为 10.5 磅 (1 磅即 1 bp), 很接近于西文基准尺寸之一的 11 pt (严格讲是 10.95 pt), 即使不再另行定义汉字字号的尺寸, 选择 11 pt 的基准尺寸, 排版输出的结果也是令人满意的. 如果一定要全部使用各个汉字字号规定的尺寸, 则需定义一系列的命令.

现有的最大字体尺寸命令是 `\Huge`, 略小于一号汉字 (28 磅). 如果需要更大的汉字, 例如初号汉字 (42 磅), 就需要自行定义一个字体尺寸命令. 新字体选择方案 (NFSS, 参见第十章) 提供了一个选择字体尺寸的命令:

$$\backslash\text{fontsize}\{\text{字体尺寸}\}\{\text{行距}\}$$

当参数不带单位时, 默认单位是 pt. 通常可取行距为字体尺寸的 1.2–1.5 倍. 如果仅是偶尔使用这种字体尺寸, 不想影响当前行距, 可将行距参数写成 `\baselineskip`. 注意上述这个字体尺寸选择命令并不会立即起作用, 其后跟上 `\selectfont` 才起作用.

使用上述命令, 将参数字体尺寸设成很大的值, 在 CJK 环境中可以得到任意大小的汉字, 但对西文, 最大只能得到 `\Huge` 对应的尺寸. 要想使用任意大小的英文字体, 可在导言区写上 `\usepackage{type1cm}`, 其中 cm 表示 Computer Modern (计

算机现代) 字体, 与厘米 (cm) 无关. 但要注意, 调用这个宏包后, 当前文档将不再使用原来标准的 fd (Font Descriptor) 文件中的定义, 生成的 dvi 文件拿到其他只安装标准 fd 文件的系统上将显示不出过大的字体.

可如下定义汉字初号命令:

```
\newcommand{\HZchuhao}{\fontsize{42bp}{\baselineskip}\selectfont}
```

这种命令都是声明, 应使用花括号限定作用范围. 本书前面的“基本篇”三个大字是如下输入的:

```
\centerline{\HZchuhao\LI 基\quad 本\quad 篇}
```

各种字体中, 即使字号相同, 字的实际面积并不相等. 对于同一字号的字, 以宋体字为标准, 隶书就显得小些, 所以使用隶书字时, 往往选用大一些的尺寸.

使用 GBK 编码时, 文稿中可以同时出现简体字和繁体字. 如果要输出繁体字, 输入也必须是繁体字, 各种冷僻字只要能输入, 就能排版输出. 這幾個字是繁體字.

下面是一个使用各种字体和字号的例子 (3-2-1.tex), 其中对应的汉字字号是近似的:

```
\documentclass[11pt]{article}
\usepackage{CJK}
\newcommand{\HZchuhao}{\fontsize{42bp}{\baselineskip}\selectfont}
\newcommand{\KAI}{\CJKfamily{kai}}
\setlength{\parindent}{0pt}
\begin{document}
\begin{CJK*}{GBK}{song}
\CJKtilde
\HZchuhao 这是初号汉字
\KAI 这是初号楷体汉字
\Huge 这是\textbf{一号}\textit{字} (\texttt{\symbol{92}Huge})
\textbf{Fermat~定理:} 对~$n\ge 3$,

$$\left\{(x,y,z)\in\mathbf{Q}^3\mid x^n+y^n=z^n,\right. \\ \left.\right\}=\emptyset.$$

\huge\CJKfamily{fs} 这是\textbf{二号}\textit{字}
(\texttt{\symbol{92}huge})
.....
\LARGE\CJKfamily{kai} 这是\textbf{三号}\textit{字}
(\texttt{\symbol{92}LARGE})
.....
\Large\CJKfamily{li} 这是\textbf{四号}\textit{字}
(\texttt{\symbol{92}Large})
.....
\large\CJKfamily{you} 这是\textbf{小四号}\textit{字}
```

```
(\texttt{\symbol{92}large})
.....
\normalsize\CJKfamily{song} 这是\textbf{五号}\textit{字}
(\texttt{\symbol{92}normalsize})
.....
\small\CJKfamily{fs} 这是\textbf{小五号}\textit{字}
(\texttt{\symbol{92}small})
.....
\footnotesize\CJKfamily{kai} 这是\textbf{六号}\textit{字}
(\texttt{\symbol{92}footnotesize})
.....
\scriptsize\CJKfamily{li} 这是\textbf{六号与七号}\textit{字}之间
(\texttt{\symbol{92}scriptsize})
.....
\tiny\CJKfamily{you} 这是\textbf{七号}\textit{字}
(\texttt{\symbol{92}tiny})
.....
\end{CJK*}
\end{document}
```

§ 3.3 居中

将较短的文本排放在一行正中间, 可以使用 T_EX 原有的命令

```
\centerline{文本}
```

使用这个命令时, 如果文本很长, 那么即使在文本中间加入了分行命令, 也不会分行, 这样排出来的行就会超出行宽. 把多行文本居中对齐排版应使用居中对齐环境

```
\begin{center}
  第一行\\
  第二行\\
  .....
  第末行\\
\end{center}
```

在上述环境中如果有某一行文本的长度超出了行宽, 它会被自动地分成几行, 分行时绝不断词, 而且每行单词之间的间隔相同, 这些行都是居中对齐的. 由于不进行断词, 还要保持一致的单词间距, 所以当文本中出现很长的单词时, 自动分行后有些行可能不等长. 对于中文, 由于任何两个字之间都允许分行, 所以自动分行后除最后一行外, 其他行通常是充满一行的.

在居中对齐环境中, 可以使用强制分行命令“\\”, 也可以使用“\\[长度]”, 以增加或减少相邻两行之间的行间隔.

在一个环境内部, 还可以使用声明

```
\centering
```

将后继文本居中对齐, 声明的作用到环境结束时为止. 但对某些超宽的行不会自动分行, 只能使用\\强制分行. 所以居中对齐声明的能力不如上述的居中环境.

如果想使文本居左或居右对齐, 可参见第 107 页.

§ 3.4 参考文献

在科技书籍和论文中常常要列举大量的参考文献, 在正文中通过文献编号进行引用. 当添加或删除一些文献后, L^AT_EX 会重新进行编号并相应更改引用的编号. 参考文献是用下列环境生成的:

```
\begin{thebibliography}{编号样本}
  \bibitem[记号]{引用标志} 文献条目
  \bibitem[记号]{引用标志} 文献条目
  ...
  \bibitem[记号]{引用标志} 文献条目
\end{thebibliography}
```

对于没有可选项记号的条目, 编译后 \bibitem 就会生成一个用方括号括起来的编号, 这些编号总是顺序排列的, 当增加或减少文献条目时, 这些编号会自动改变. 由于 T_EX 在第一次编译时, 先遇到引用, 最后才编译文献目录, 因此第一次编译遇到引用时只能用问号“?”代替文献编号, 还需要再编译一次, 才能去掉问号, 代以正确的编号.

引用标志不可省略, 它不会显示在排印结果中. 在正文中, 使用

```
\cite{引用标志1, 引用标志2, ...}
```

引用相应的一些文献, 在排印结果中, 上述引用命令就变成了被方括号括起来的文献编号. 引用标志可以用字母、数字和除了逗号外的符号组成.

真正的文献信息写在文献条目中, 包含作者、题目、出版社、年代、版本、页码等内容, 不同部分一般使用不同的字体. 如果一行排不下, 后面的行会向右缩进, 缩进的距离等于编号样本的宽度.

编号样本可以是一个数字, 它的位数应是参考文献最大编号的位数, 以使缩进的文本能够对齐. 如果使用了[记号], 该文献不会被编号, 在相当于编号的位置显示记号本身, 在正文中通过引用标志引用, 排版后显示为给定的记号. 当记号的宽度大于编号的宽度时, 应在编号样本处写最宽的记号.

这个环境实际上是一个 list 环境 (参见第 123 页), 条目之间的间隔通常会大于正文中的行间隔, 在环境内部使用命令

```
\setlength{\itemsep}{长度}
```

可以改变条目之间的竖直间距, 若将其中的参数 长度 设为 0pt, 则条目之间的间隔就与正文的行间隔相同了。

在 article 文档类的参考文献环境中, 默认标题采用英文字 “References”, 所用字体为 “\Large\bfseries”, 而且向左对齐 (系统内部在标题右面使用了命令 \hfil). 如果想采用中文标题或别的西文标题, 可把以下命令插在参考文献环境之前:

```
\renewcommand{\refname}{\hfil\HEI 参\quad考\quad文\quad献}
```

前面加上命令 \hfil 是为了按照中文的习惯, 让 “参考文献” 居中. 注意这里不能用 \hfill, 否则抵消了内部原有的 \hfil, 就变成标题靠右了. 命令 \HEI 的定义见第 23 页.

如果用的是 book 或 report 文档类, 则参考文献环境中默认标题采用英文字 “Bibliography”, 如果想采用中文标题或别的西文标题, 可把以下命令插在参考文献环境之前:

```
\renewcommand{\bibname}{\hfil 参~考~文~献}
```

下面的例子给出了引用及参考文献环境的用法, 完整的源文件参见 3-4-1.tex.

```
% 3-4-1.tex
关于其中细节, 请参见~\cite{lamport, knuth, dbk}.
\renewcommand{\refname}{\hfil\HEI 参\quad考\quad文\quad献}
\begin{thebibliography}{Lam}
\bibitem[Lam]{lamport}Lamport, L. \textsl{\LaTeX} --- A
Document ...
\bibitem[Knu]{knuth}Knuth, D. E., \textsl{The \TeX}book},
...
\bibitem[dbk]{中国大百科全书}
\end{thebibliography}
```

其打印输出如图 3.1 所示.

§ 3.5 制表位

使用 Word 时, 可以在一行上设置几个制表位, 每当按一下 tab 键, 光标就跳到下一个制表位, 这对于排列竖向对齐的几列文字是相当方便的. 在 TeX 中, 用

关于其中细节, 请参见 [Lam, Knu, 1].

参 考 文 献

[Lam] Lamport, L. *ℒ_AT_EX – A Document ...*

[Knu] Knuth, D. E., *The T_EXbook*, ...

[1] 中国大百科全书

图 3.1 参考文献输出示例

tabbing 环境实现了类似功能: 左边界自动是一个制表位 (称为第零个制表位), 然后在一行中任何地方使用设位命令 `\=`, 该处就成为一个制表位, 在后面的行中, 跳格命令 `\>` 表示排版时跳到下一个制表位. 第零个制表位是默认使用的, 不需要跳格命令. 每一行都用 `\>` 结束.

设置制表位时, 为了精确控制列宽, 可以使用 `\hspace` 命令, 或者利用后面出现的各列中最宽的项设置制表位, 这样的行称为样本行, 它的作用是仅仅设置制表位, 而不被显示出来, 规定这种样本行末尾不能使用 `\>`, 而必须用命令 `\kill` 结束.

第 69 页的表格就是使用 tabbing 环境 如下输入的:

```
{\CJKfamily{kai}\ttfamily
\begin{tabbing}
\hspace*{4em}\=article\quad \=文章类\kill
\>book          \>书籍类\\
\>report         \>报告类\\
\>article        \>文章类\\
\>letter         \>书信类
\end{tabbing}
}
```

在每一行中, 随时可用设位命令 `\=` 重设或添加 (不是插入) 制表位. 如果在一行中有足够的跳格命令, 使最后一个跳格命令正好跳到最后一个制表位, 则可在该行最后一个跳格命令之后的任何位置用 `\=` 添加一个新的制表位. 如果在中间某个 (例如第二个) 制表位之后加入了设位命令 `\=`, 则是重设该制表位的下一个 (即第三个) 制表位的位置, 应使重设的制表位不超过再下一个 (第四个) 制表位的位置, 否则会排版成出乎意料的结果.

下面是重设和添加制表位的例子, 上为输入, 下为输出.

```

\begin{tabbing}
这是第一列\quad \= 这是第二列 \\\
左列 \> 中列\quad \= 新添第三列 \\\
新一列\quad\= 新二列 \> 对齐第三列\\
列1 \> 列2 \> 列3
\end{tabbing}

```

```

这是第一列 这是第二列
左列      中列  新添第三列
新一列  新二列  对齐第三列
列1      列2      列3

```

关于 tabbing 环境的更深入的介绍, 可参见提高篇。

几点说明:

1. $\text{T}_{\text{E}}\text{X}$ 像处理段落一样处理 tabbing 环境, 也就是说, 必要时会自动在两行间分页。该环境中不能使用分页命令。如果一定要在某处强制分页, 可使用一个变通的方法: 在想要分页的行末加上一个充分大的行间隔, 例如 `\\[15cm]`, 排版输出时, 这个间隔超出了当前页的剩余空间, 于是就在此处分页了, 又因为在 `\\` 后面未加 `*` 号, 所以下一页顶部不会出现空白。顺便提及, 后面将要介绍的 `tabular` 环境不会自动分页。

2. 在 WORD 中, 当输入的文本很长时, 可能占用几个制表位, 按 `tab` 键会跳到后面还未占用的制表位上。tabbing 环境与此不同, 如果前一个跳格命令跳到第 n 个制表位上, 则无论它后面的文本多长, 下一个跳格命令总是跳到第 $n+1$ 个制表位上, 这实际上可能产生回退, 造成文字重叠。此外如果一行中的跳格命令超过了制表位的个数, 编译时会显示出错信息。

3. tabbing 环境不会自动分行, 只有在遇到 `\\` 时才另起一行, 所以输入的文本不可太长。

4. 在 tabbing 环境中, 放在一行中的字体命令只对该行起作用, 不影响其他行。此外与弹性长度 `\fill` 有关的命令没有作用。

5. 关于制表位的深入技巧, 请参看提高篇第 114 页。

§ 3.6 表格

3.6.1 表格环境

构造表格可使用环境:


```
\begin{tabular}[竖向位置]{列格式}
  第一行\\
  第二行\\
  ..... \\
  第末行
\end{tabular}
```

表格环境就是创建一个小页. 参数的名字已大致表明了其含义, 下面逐个详细说明.

竖向位置 确定表格在竖直方向上与当前外部文本行的相对位置, 默认(即无该可选项时)表格相对于外部基线居中排放. 可取值为:

- t 表格顶部基线与外部基线对齐, 顶部可能是文本, 也可能是横线;
- b 表格底部基线与外部基线对齐, 底部可能是文本, 也可能是横线.

列格式 指定表格各列的格式. 列格式由若干项组成, 表格的每一列对应于列格式中的一项, 此外还可能含有对应于表格边界和列间分隔线的一些项. 对应于列的项可以是下列值:

- l 对应的列的内容靠左对齐;
- c 对应的列的内容居中对齐;
- r 对应的列的内容靠右对齐.

对应于边界或列间分隔线的格式项有:

- | 画单条竖直线;
- || 画双条竖直线(两条相距很近的竖直线).

行(表格行) 表格中的每一行都是由若干列组成, 输入时相邻列之间用符号&隔开, 列的内容可以是空的, 但列的分隔符&符号不能省略, 除非从某一列开始其后各列都是空白而且不画边界竖线, 该列之后的&符号才可以省掉.

对应于表格每一行的输入内容都必须用\\结束.

T_EX把每一列的内容当成一个组, 即相当于用花括号把列的内容括起来了, 因此放在列文本中的一些命令如字体字号声明, 其作用将被局限于该行该列之内.

下面是一个把相邻几列当作一列使用的命令:

`\multicolumn{列数}{列格式}{文本}` 该命令把接下来的指定个数的列组合成单个列, 其宽度等于各列的总宽度, 包括属于该列两侧的列间隔(相邻两列间的空白间隔一半属于左列, 一半属于右列). 列格式指定组合后的

单个列的排放格式, 其中必须包含一个 (而且只能包含一个) 定位 参数 l、c 或 r, 可以同时包含竖线 “|”. 当指定的列数是 1 时, 该命令可以改变当前行当前列的对齐方式.

下面是几个在表格中画横线或竖线的命令:

`\hline` 画一条与表格同宽的水平横线. 隐含换行符, 即画线后会自动换行, 所以在该命令后面不要写 `\\`. 该命令只能出现在一行的最前面或行结束符 `\\` 的后面, 它在刚结束的行的下面画一条横线. 连用两次该命令, 会画出间距很小的两条横线.

`\cline{m-n}` 从第 m 列的开始位置画一条水平横线到第 n 列的结束位置. 该命令必须位于行结束符 `\\` 的后面, 可以多次出现. 命令 `\cline{2-4}` `\cline{6-9}` 会在刚结束的行的下面画左右两条横线, 一条是从第 2 列到第 4 列, 另一条是从第 6 列到第 9 列. 该命令使用全部列宽, 即包括指定列所属的左右列间隔空白宽度.

`\vline` 画一条竖直线, 其高度等于所在位置的行高. 放在表格环境开始的列格式中的竖线会画出贯穿整个表格的竖直线, 而这里的命令只画出等于行高的竖直线.

3.6.2 样例

输入简单的表格, 只要看看下面的例子就够了, 只有在设计复杂表格时, 才需要了解上一节或提高篇第 115 页有关的知识.

例: 这是两个内容相同的表格, 只是字体和表格线有所不同:

姓名	住址	电话
张爱国	中山路 3 号	12345678
王自强	南京西路 10 号	87654321
李立	北京中路 345 号 501 室	

姓名	住址	电话
张爱国	中山路 3 号	12345678
王自强	南京西路 10 号	87654321
李立	北京中路 345 号 501 室	

上面第一个表格输入如下:

```
\begin{tabular}{l|l|l}
姓名 & 住址 & 电话\\
\hline
张爱国 & 中山路~3~号 & 12345678 \\
王自强 & 南京西路~10~号 & 87654321 \\
```

```
李立 & 北京中路~345~号~501~室
\end{tabular}
```

上面第二个表格输入如下:

```
\newcommand*{\HEI}{\CJKfamily{hei}}
\begin{tabular}{|l|l|l|}
\hline
\HEI 姓名 & \HEI 住址 & & \HEI 电话\\\hline
张爱国 & & 中山路~3~号 & & 12345678 \\\hline
王自强 & & 南京西路~10~号 & & 87654321 \\\hline
李\quad立 & & 北京中路~345~号~501~室 & & \\\hline
\end{tabular}
```

输入时在原稿上不需要对齐各项,但为了便于检查错误,输入时稍稍对齐一些还是有好处的. 另外注意上一表格输入时省略了一个&符号,而下一表格就不能省掉这个&符号,否则右边框线就会缺少一段连不起来了. 此外要注意,位于单元格中的字体命令只在该格内起作用.

如果我们再看一下上面的两个表,会发现由于第一行的栏目标题不居中,影响了美观. 这可利用\multicolumn来解决. 此外,为了使单名与双名能对齐,我们可在单名中间插入一个全角的空格或一个\quad. 全角的空格等于空出一个汉字的位置,可用于汉字的对齐,能起到西文空格无法起到的作用. 此外,为了使表格的标题不会因换页而与表身分离,我们可以把标题也当作表格的一行来处理,并利用命令\\[5pt]使其与表身隔开. 现在我们把上述表格的输入修改如下:

```
\begin{tabular}{|l|l|l|}
\multicolumn{3}{|c|}{\large\HEI通\quad讯\quad录}\\[5pt]
\hline
\multicolumn{1}{|c|}{\HEI姓名}
&\multicolumn{1}{|c|}{\HEI住\quad址}
&\multicolumn{1}{|c|}{\HEI电 话}\\\hline
张爱国 & & 中山路~3~号 & & 12345678 \\\hline
王自强 & & 南京西路~10~号 & & 87654321 \\\hline
李 立 & & 北京中路~345~号~501~室 & & \\\hline
\end{tabular}
```

排版结果如下:

通 讯 录

姓名	住 址	电 话
张爱国	中山路 3 号	12345678
王自强	南京西路 10 号	87654321
李 立	北京中路 345 号 501 室	

关于表格更深入的技巧和样例, 请参看提高篇第 115 页.

§ 3.7 脚注

自动编号的脚注命令是

`\footnote{脚注文本}`

该命令应紧接在需要注释的文字后面, 排版输出时会在该处显示一个脚注标记, 并用较小的字体 (`\footnotesize`) 以脚注形式将 脚注文本 显示在当前页的底部, 并带有同样的脚注标记. 脚注的首行自动缩排, 在一页上的第一个脚注与正文之间有一条水平短线把正文与脚注分隔开来.

脚注的标记是一个上标形式的数字, 它是自动顺序编号的.

本页的脚注¹是如下产生的: 在“本页的脚注”和“是如下产生的”之间插入了命令

```
\footnote{这是一个脚注(footnote)例子}
```

在 `article` 类中, 整个文档对自动编号的脚注统一编号. 在 `report` 和 `book` 文档类, 只在一章内统一编号, 每当开始新的一章, 自动编号的脚注都重新从 1 开始计数.

脚注只能位于通常的段落模式中, 不能位于数学模式、表格、LR 盒子或子段盒子 `\parbox` 中. 但可以位于小页环境中, 此时脚注显示在小页的底部.

关于脚注的更深入的技巧, 请参看提高篇第 126 页.

¹这是一个脚注(footnote)例子

第四章 数学公式

前面几章介绍的是正文的排版(称为文字模式),本章介绍数学公式的排版(数学模式),T_EX 最具竞争力的特点就是能够排版输出精美的数学公式.

为了能充分使用 L^AT_EX 提供的数学符号和数学粗体字体,可在源文件导言区放上下列命令

```
\usepackage{latexsym,bm}
```

以加载宏包 latexsym 和 bm.

§ 4.1 概述

文本模式和数学模式对排版的要求是不同的,最简单的例子就是在文本模式中通常将字母排成正体,空格可以分隔单词,而在数学模式中则将表示变量的字母排成斜体,空格都被吃掉,T_EX 会自动安排公式各部分之间的间距.

为了标明源文件中某段内容是数学公式,必须在该段内容的两边加上特殊标记,以“通知”T_EX 系统对标记的内容按数学模式进行排版处理.需要特别注意的是这种标记总是“成对”出现的,前一个标记表示进入数学模式,后一个标记表示退出数学模式,前后标记的不匹配是造成 T_EX 排版错误的常见原因.

根据数学公式出现的不同位置,将它们分为两类:出现在一行文字内部的称为行内公式,亦称正文公式(text formulas);出现在两行之间的称为行间公式,亦称显示公式(displayed formulas).

在数学模式中有 4 个基本命令控制字体的大小式样,它们是:

```
\displaystyle    \textstyle  
\scriptstyle    \scriptscriptstyle
```

可分别称为“显示式样”,“正文式样”,“角标式样”和“二级角标式样”.T_EX 会根据不同情况自动调用合适的式样命令,用户也可以使用这些命令强制改变字体大小.

在数学公式中出现的下述数学符号

+ - / = < > () [] | ' ! :

都可直接从键盘输入,但是数学公式中的花括号(亦称曲括号或大括号)不能直接输入,而必须输入成`\{`和`\}`,这是因为不带前导斜线的花括号是 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 的分组(集团)标记,排版后不会被显示出来.此外注意,如果冒号不作为数学符号,而仅是作为标点符号,则不要直接用符号“:”,而应用命令`\colon`,否则在输出的冒号前面可能产生多余的空白间隔.

数学模式中省略号有如下4种:








输出字符	⋮	⋱
输入序列	<code>\ldots</code>	<code>\cdots</code>	<code>\vdots</code>	<code>\ddots</code>

两个水平的省略号的区别是输出位置不同,一个靠下,另一个上下居中.其他两个多用于矩阵,但没有形如 \ddots 的上升省略号,如果要用的话,需自己定义,例如定义为:

```
\newcommand*{\adots}{\mathinner{\mkern2mu%
\raisebox{0.1em}{.}\mkern2mu\raisebox{0.4em}{.}%
\mkern2mu\raisebox{0.7em}{.}\mkern1mu}}
```

其中`\mkern`是用于在数学环境中插入间隔,`mu`是用于数学环境中的长度单位,其长度等于`em`的 $1/18$.

在数学模式中,输入的空格不会输出空白,如果要插入空白间隔,可使用下列控制命令:

<code>\quad</code>	= 两个空铅(quad)宽度	
<code>\quad</code>	= 一个空铅宽度	
<code>\;</code>	= 5/18 空铅宽度	
<code>\:</code>	= 4/18 空铅宽度	
<code>\,</code>	= 3/18 空铅宽度	
<code>\!</code>	= -3/18 空铅宽度	
<code>\ </code>	= 一个空格	
<code>\hspace{长度}</code>		
<code></code>		

上述这些命令除了`\;`,`\:`和`\!`只能用于数学模式外,其他命令既可以用于数学模式也可以用于文字模式.其中`\!`为负的空白,实际是缩短间隔.命令`\hspace`和`\phantom`的解释见2.6.1节.

在数学模式中,字母或文字被看成是变量的名字,显示为数学斜体(`\mathit`),空格均被吃掉不起作用,例如输入`$This is a word$`,显示结果是 $This is a word$.在数学模式中,需要显示普通文本时,应把文本放在LR盒子中,即输入成`\mbox{文本}`,其中文本既可以是西文,也可以是汉字.盒子内部的空格是起作用的,但在盒子与数学公式之间,空格不起作用,仍需用上述的一些命令插入适当的间隔.

§ 4.2 行内公式

出现在一行之内的数学公式称为行内公式, 例如“勾股定理 $a^2 + b^2 = c^2$ 也称商高定理”这句话中出现的数学公式就是行内公式.

L^AT_EX 提供的行内公式的标记是 `\begin{math}` 和 `\end{math}`, 分别表示行内公式的开始和结束, 也可以使用简化的标记 `\(` 和 `\)`, 或者更简单的使用 T_EX 本来的标记——开始和结束标记都写作 `$`. 例如下列 3 种输入

勾股定理 `\begin{math} a^2+b^2=c^2 \end{math}` 也称商高定理

勾股定理 `\(a^2+b^2=c^2 \)` 也称商高定理

勾股定理 `$ a^2+b^2=c^2 $` 也称商高定理

产生同样输出:

勾股定理 $a^2 + b^2 = c^2$ 也称商高定理

在 L^AT_EX 中将形如

`\begin{...}`

.....

`\end{...}`

的排版标记称为“环境”. 对于很短的公式, 大多数人喜欢使用简化的标记, 对于很长的行内公式建议使用环境标记, 并将环境的开始和结束标记分别单独写在一行如下面的样子以便于检查错误和修改文件:

勾股定理
`\begin{math}`
 $a^2+b^2=c^2$
`\end{math}`
 也称商高定理

§ 4.3 行间公式

位于两行之间的公式称为行间公式, 例如下面的公式

$$\int_a^b f(x) dx = \lim_{\|\Delta x_i\| \rightarrow 0} \sum_i f(\xi_i) \Delta x_i$$

就是行间公式. 行间公式有多种情况, 例如公式可能只有一行, 也可能有多行; 行间公式可以带编号, 也可没有编号; 带编号时既可以用人工编号, 也可以由 L^AT_EX 自动

编号 (这是 L^AT_EX 的主要优点之一). 我们以表格形式列出在各种情况下 L^AT_EX 所使用的行间公式的标记.

	无编号	自动编号	人工编号
单行公式	$\begin{displaymath}$ (公式内容) $\end{displaymath}$	$\begin{equation}$ (公式内容) $\end{equation}$	$\$ \$$ 公式 $\backslash eqno$ 编号 $\$ \$$ 或 $\$ \$$ 公式 $\backslash leqno$ 编号 $\$ \$$
多行公式	$\begin{eqnarray*}$ (公式内容) $\end{eqnarray*}$	$\begin{eqnarray}$ (公式内容) $\end{eqnarray}$	较复杂, 见后面章节

注意, 无编号和自动编号的多行公式的环境仅相差一个星号, 但单行公式的两个环境所用文字是很不同的.

类似于行内公式的 math 环境, 行间公式的 displaymath 环境也有简化形式:

$\backslash [\dots \backslash]$

或者使用 T_EX 原有的行间公式标记

$\$ \$ \dots \$ \$$

注意, 当使用 $\$$ 作为数学模式的标记时, 在行内公式两端用的是单个的 $\$$, 而在行间公式两端使用的是紧连在一起的双 $\$$.

对于自动编号的公式, 也可以人工干预 (改变) 编号, 或指定多行公式中的某行公式不放编号, 在后面几节中会给出相应的例子. 进一步讨论可参见提高篇. 如果不想利用 L^AT_EX 的自动编号功能, 而由人工直接给出编号, 可以使用 T_EX 中原有的格式:

$\$ \$$ 数学公式 $\backslash eqno$ 编号 $\$ \$$

或

$\$ \$$ 数学公式 $\backslash leqno$ 编号 $\$ \$$

其中 $\backslash eqno$ 将编号放在公式的右面, $\backslash leqno$ 将编号放在公式的左面. 对于这种带有人工编号的公式, 不能将 $\$ \$ \dots \$ \$$ 换成 $\backslash [\dots \backslash]$.

例如输入

$\$ \$ a^2+b^2=c^2, \backslash eqno(**) \$ \$$
由公式 ($\$ \$$) 就可得到所需结论.

可得到以下输出:

$a^2 + b^2 = c^2,$ (**)
由公式 (**) 就可得到所需结论.

输入数学公式时, 数学环境前后的空行仍表示分段. 如果环境后面没有空行, 则行间公式下方的文字不会缩进, 即不形成新的段落.

§ 4.4 上标和下标

上标和下标统称角标, $\text{T}_{\text{E}}\text{X}$ 系统会用较小的字体排版角标, 并升高上标或降低下标的位置. 上标命令是 $\text{\textasciitilde}\{\dots\}$, 下标命令是 $\text{_}\{\dots\}$; 当角标是单个字符时可不用花括号. 在上一节的例子中已见到上标的打法, 下面再举几例. 输入

```
\[
  x_1, x_1^2, x^{2_1}, x_{22}^{(n)}, {}^*\!x^*
```

排版输出为

$$x_1, x_1^2, x^{2_1}, x_{22}^{(n)}, {}^*x^*$$

从上例可见, 当有上标和下标时, 输入次序是无关紧要的, x_1^2 和 x^{2_1} 输出同样结果 x_1^2 . 另外可见, 公式中间输入的空格不起作用. 本例最后一个例子给出了在一个变量左右两边打印角标的一种方法.

输入多层角标时, 必须注意分组括号的使用, 输入

```
\[
  x^{\{y^z\}}, \quad x^{\{y_z\}}, \quad \quad x^{\{x^{\{x^x\}}\}}
```

输出为

$$x^{y^z}, \quad x^{y_z}, \quad \quad x^{x^{x^x}}$$

输入时若漏掉分组括号, 比如把 $x^{\{y^z\}}$ 输入成 $x^y{}^z$, 则会出现编译错误:

! Double superscript.

如果忽略这个错误, 那么输出结果是 x^{yz} 而不是 x^{y^z} .

可以看出一级角标字体变小了, 二级角标字体更小, 但更高级的角标不再变化, 都和二级角标一样大. 若当前字体大小是 10pt, 则一级角标为 7pt, 二级角标为 5pt, 因此当使用文字作上下标时, 应加上改变大小的命令, 并且不要忘记将它们括在 $\text{\mbox}\{\}$ 内. 在 $\text{\mbox}\{\}$ 内是文字模式, 应该用文字模式下的字号命令. 例如输入

```
\[
  S_{\text{\mbox{\scriptsize外}}}, \quad S_{V_{\text{\mbox{\tiny极大}}}}]
```

输出为

$$S_{\text{外}}, S_{V_{\text{极大}}}$$

当角标位置看起来不明显时, 可以强制改变角标字体的大小或角标的层次(如把一级角标改为二级角标). 下例是变量 y 带有大写字母下标 N , 不同输入产生的不同输出. 输入

```
\[
  y_N,\quad y_{_N},\quad y_{-\{\scriptstyle N\}}
\]
```

输出为

$$y_N, \quad y_N, \quad y_N$$

第一种输出结果几乎看不出是下标, 第二种是把一级下标改为二级下标(字体自动变为二级角标大小), 第三种也是把一级下标改为二级下标, 但强制字体为一级角标的大小, 后两种显然比第一种好看得多.

求导符号的撇号可用控制命令 `\prime`, 由于撇号应出现在上标位置, 所以需要用上标命令. 或者直接用键盘符号 `'` 代替 `\prime`, 但因键盘上的撇号(单引号)显示的位置已经在上标的位置了, 因此对它就不要再用上标命令了. 例如输入 `$f^{\prime\prime}$` 或 `f''`, 输出都是 f'' .

排版双行或多行下标见第 54 页和第 139 页.

§ 4.5 分式

当输入较短的分式时, 最简单的方法是使用斜线, 如输入 `$(x+y)/2$` 产生输出 $(x+y)/2$. 但本节所讲的分式是指带有水平分数线的分式, 所用命令为

```
\frac{分子}{分母}
```

当分子或分母是单个字符时, 可不用花括号括起来. 下面是一些例子.
输入行内分式

```
分式 $\frac{x+y}{2}$ 和 $\frac{1}{1+\frac{1}{2}}$.
```

输出为

```
分式  $\frac{x+y}{2}$  和  $\frac{1}{1+\frac{1}{2}}$ .
```

输入行间分式

```
\[
  \frac{x+y}{2}, \quad \frac{1}{1+\frac{1}{2}}
\]
```

则输出

$$\frac{x+y}{2}, \quad \frac{1}{1+\frac{1}{2}}$$

行内分式中的字体显得比行间分式小, 这是因为行内分式使用的是角标字体, 分式中的分式使用更小一级的字体(到二级角标字体为止, 不会再小了). 可以人工改变分子和分母所用字体的大小, 见下例的输入和输出, 其中定义一个控制命令是为了简化输入:

```
\newcommand{\DF}[2]{\displaystyle\frac{#1}{#2}}
```

输入行内公式 $\$ \backslash DF\{x+y\}2 \$$ 输出 $\frac{x+y}{2}$

对于连分式, 所有分子和分母具有同样大小时比较好看, 对比下面两例: 输入

```
\[
  a_0+\frac{1}{a_1}
    +\frac{1}{a_2}
    +\frac{1}{a_3}
    +\frac{1}{a_4}}
\]
```

输出

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

输入

```
\[
  a_0+\DF{1}{a_1}
    +\DF{1}{a_2}
    +\DF{1}{a_3}
    +\DF{1}{a_4}}
\]
```

输出

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

分数线长度的预设值是分子或分母所占的最大宽度, 如果想使分数线变长些, 可在分子或分母两侧添加一些间隔. 在中文文档中, 习惯使用略长的分数线, 可为此定义一个中文分式命令

```
\newcommand{\zwfs}[2]{\frac{\;#1\;}{\;#2\;}}
```

请观察下例中的输入和输出. 输入

```
\[
  \frac{1}{2}\quad \zwfs{1}{2}
\]
```

输出 (对比分数线长度)

$$\frac{1}{2} \quad \frac{1}{2}$$

§ 4.6 根式

打印根式的命令是

```
开平方: \sqrt{表达式}
开高次方: \sqrt[n]{表达式}
```

其中 n 是开方次数. 当被开方表达式是单个字符时可以用不用花括号, 若是单个字母, 不用花括号时则需用空格与 `\sqrt` 隔开. 根式可以嵌套. 下面举几个例子.

输入

```
$_\sqrt{2}<\sqrt{3}3$
```

输出

$$\sqrt{2} < \sqrt[3]{3}$$

输入

```
\[
  \sqrt{a}+\sqrt{b}+\sqrt{c},\quad
  \sqrt{1+\sqrt[p]{1+\sqrt[q]{1+a^2}}}\]

```

输出

$$\sqrt{a} + \sqrt{b} + \sqrt{c}, \quad \sqrt{1 + \sqrt[p]{1 + \sqrt[q]{1 + a^2}}}$$

从上面的例子可以看出两个问题, 一是当被开方表达式是高度不同的字母时 (如上例中的 a 和 b), 根号上面的横线不在同一水平线上. 为了克服这一缺点, 可以在被开方表达式中插入一个只有高度没有宽度的“数学支柱” (`\mathstrut`), 将被开方表达式支撑到同一高度, 根号上面的横线就位于同一水平线上了, 如下例所示, 输入

```
\[
  \sqrt{\mathstrut a} + \sqrt{\mathstrut b}
  + \sqrt{\mathstrut c}\]

```

输出为

$$\sqrt{a} + \sqrt{b} + \sqrt{c}$$

`\mathstrut` 的宽度为零, 高度和深度与圆括号相同.

第二个问题是当被开方表达式较高时, 开方次数的位置显得低了些. 一个简单但不完美的调整办法是把开方次数变为上标, 并拉近与根号的水平距离, 即将开方命令中的 $[n]$ 改为 $^n\!$, 如下所示, 输入

```
\[
  \sqrt{1+\sqrt[^p\!]{1+\sqrt[^q\!]{1+a^2}}}\]

```

输出为

$$\sqrt{1 + \sqrt[p]{1 + \sqrt[q]{1 + a^2}}}$$

若不要根号上面的横线, 则不要使用 `\root`, 而应改用 `\surd`. 例如输入

```
\[
\surd (x+y+z)
\]
```

输出为

$$\sqrt{(x+y+z)}$$

§ 4.7 求和、积分

产生求和号和积分号的命令分别是 `\sum` 和 `\int`, 它们有一些共同的特点: 一是通常都带有上下限 (与上下标的输入方法相同), 二是符号的大小在行内公式和行间公式中是不同的, 被称为是“具有两种尺寸的数学符号”. 对于求和号, 上下限的位置也会变化. 例如输入

```
在行内公式中的求和号与积分号: $\sum_{k=1}^n, \int_a^b$
在行间公式中的求和号与积分号: \[\sum_{k=1}^n, \int_a^b\]
```

输出为

```
在行内公式中的求和号与积分号:  $\sum_{k=1}^n, \int_a^b$ 
在行间公式中的求和号与积分号:
```

$$\sum_{k=1}^n, \int_a^b$$

在带有上下限的符号后面紧跟着写上 `\limits`, 然后再写上下限, 就会使上下限的位置移到符号的头顶和脚下. 输入如下内容

```
在行内公式中的求和号与积分号 (注意上下限位置):
$\sum\limits_{k=1}^n, \int\limits_a^b$
```

输出为

```
在行内公式中的求和号与积分号 (注意上下限位置):  $\sum_{k=1}^n, \int_a^b$ 
```

与 `\limits` 对应的命令 `\nolimits` 则总是强制上下限位于符号的右侧, 例如输入

```
\[
  \sum_{i=1}^n a_i \quad \sum\nolimits_{i=1}^n a_i
\]
```

输出为

$$\sum_{i=1}^n a_i \quad \sum_{i=1}^n a_i$$

在输入完整的积分公式时, 有两个细节需要注意, 一是被积表达式与其后的微分算子之间最好有一点小的间隔, 即放入小间隔控制符 “\,”; 二是微分算符 “d” 应是直体, 即把它输入成 `\mathrm{d}`。对比下例输出的两个积分公式的细微区别。输入

```
\[
  \int_a^b f(x)dx \quad \int_a^b f(x)\,,\mathrm{d}x
\]
```

输出为

$$\int_a^b f(x)dx \quad \int_a^b f(x) \, \mathrm{d}x$$

国内有些出版社要求使用直立的积分号, 为此可有两种解决方法。第一种方法是使用宏包 `wasysym`, 即在导言区的尾部 (至少应在调入 `amsmath` 的命令之后) 插入命令 `\usepackage{wasysym}`, 然后用 `\varint` 代替 `\int`, 就可得到直立的积分号。与此同时, 命令 `\iint`, `\iiint`, `\varoint`, `\varoiint` 得到的积分号都是直立的。请读者参看例 4-7-1.tex。如果编译时出错, 请先检查你的 MiKTeX 系统有没有安装宏包 `wasysym`。

另一种方法是在导言区插入以下命令:

```
\DeclareSymbolFont{euex}{U}{euex}{m}{n}
\DeclareMathSymbol{\varint}{\mathop}{euex}{"52}
\DeclareMathSymbol{\varoint}{\mathop}{euex}{"48}
```

在数学模式内使用命令 `\varint` 与 `\varoint` 也可得到直立的积分号。参看例 4-7-2.tex。

§ 4.8 数学重音符号

数学模式中的重音符号在用途与打法上都不同于文字模式的重音符号 (参见第 14 页), 数学中的重音符号通常用于区分同一个字母表示的不同的变量, 输入方法如下:

\hat{a}	<code>\hat{a}</code>	\check{a}	<code>\check{a}</code>	\breve{a}	<code>\breve{a}</code>
\tilde{a}	<code>\tilde{a}</code>	\bar{a}	<code>\bar{a}</code>	\vec{a}	<code>\vec{a}</code>
\acute{a}	<code>\acute{a}</code>	\grave{a}	<code>\grave{a}</code>	\mathring{a}	<code>\mathring{a}</code>
\dot{a}	<code>\dot{a}</code>	\ddot{a}	<code>\ddot{a}</code>		

在数学模式中当字母 i 和 j 带有重音符号时, 应去掉顶上的小点, 此时这两个字母需改用 `\imath` 和 `\jmath`, 例如为了输出 \hat{i} 与 \breve{j} , 应输入 `$(\hat{\imath})(\breve{\jmath})$`。

符号 `\hat` 和 `\tilde` 有“宽型”版本 `\widehat` 和 `\widetilde`, 它们可伸展到 3 到 4 个字母的宽度, 例如输入

```
\[
\widehat{ab} + \widehat{cdef} = \widetilde{xyz}
\]
```

输出为

$$\widehat{ab} + \widehat{cdef} = \widetilde{xyz}$$

§ 4.9 上画线、下画线及类似符号

在公式的上方和下方画直线的命令分别是

```
\overline{公式}    \underline{公式}
```

在公式的上方和下方画花括号的命令分别是

```
\overbrace{公式}    \underbrace{公式}
```

看几个例子, 输入

```
\[
\overline{\overline{a}^2 + \underline{ab} + \bar{b}^2}
\]
```

输出为

$$\overline{\overline{a}^2 + \underline{ab} + \bar{b}^2}$$

输入

```
\[
  \underbrace{a + \overbrace{b + \dots + b}^{m\uparrow}}_{20\uparrow} + c
\]
```

输出为

$$\underbrace{a + \overbrace{b + \dots + b}^{m\uparrow} + c}_{20\uparrow}$$

从例子中可以看出对于单个字符使用 `\overline` 比使用 `\bar` 得到的线要稍长一些。此外,用于上下花括号的角标总是出现在花括号的正中上方或下方,而不是通常的角标位置。

§ 4.10 堆叠符号

当需要把一个符号堆叠在另一个符号的上方时,可以使用命令

```
\stackrel{上位符号}{基位符号}
```

其中基位符号会被用正常字体打印在正常位置,而上位符号则是用较小的字体打印在基位符号的上方,例如输入

```
\[
  \vec{x} \stackrel{\mathrm{def}}{=} (x_1, \dots, x_n)
\]
```

输出为

$$\vec{x} \stackrel{\mathrm{def}}{=} (x_1, \dots, x_n)$$

利用控制数学字体大小的命令可以使堆叠的上下符号具有同样大小,对比下例中的三个“小于或等于”符号。输入

```
\[
  A \leq B
  \stackrel{<}{=} C
  \stackrel{\textstyle<}{=} D
\]
```

输出为

$$A \leq B \leq C \leq D$$

其中第一个是使用现成的命令`\le`, 第二个是使用堆叠命令构造的小于等于号, 第三个也是使用堆叠命令构造的新符号, 但将上位符号的大小改成了与基位符号大小相同。

`\stackrel`命令的两个参数处于不平等地位, 如果想打印一个类似于分式但不带分数线的公式, 就不能使用这个命令了, 此时可以使用后面将要介绍的`\begin{array}`环境, 或使用 $\mathrm{T}_\mathrm{E}\mathrm{X}$ 原有的数学命令:

```
{上位公式 \atop 下位公式}
{上位公式 \choose 下位公式}
```

这两个命令中的上位公式与下位公式地位是平等的, 使用同样大小的字体, 它们都生成一个类似于没有分数线的分式结构。其中`\choose`生成的整个公式被包围在圆括号内, 是二项式系数的一种表示方法。例如输入

```
\[
{n+1 \choose k} = {n \choose k} + {n \choose k-1}
\]
```

输出为

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$$

使用`\atop`命令打印双行下标很方便, 例如输入

```
\[
\sum_{k_0,k_1,\ldots>0 \atop k_0+k_1+\cdots=n}
A_{0k_0}A_{1k_1}\cdots
\]
```

输出为

$$\sum_{\substack{k_0,k_1,\ldots>0 \\ k_0+k_1+\cdots=n}} A_{0k_0}A_{1k_1}\cdots$$

上例还演示了两种水平省略号的用法, 在逗号后用的是 `\ldots`, 在加号后用的是 `\cdots`. 一般来说, 水平省略号应与它前面的整体符号的中部对齐, 由此决定使用哪一种省略号.

若排版多行下标, 见第 139 页.

§ 4.11 可以变大的定界符

这里所谓的定界符是指包围或分割公式的一些符号, 如

(())	[\lfloor]	\rfloor
[[]]	[\lceil]	\rceil
{	\{	}	\}	<	\angle	>	\rangle
			\	↑	\uparrow	↑	\Uparrow
/	/	\	\backslash	↓	\downarrow	↓	\Downarrow
				↑↓	\updownarrow	↑↓	\Updownarrow

在上述符号前加上命令 `\big`、`\Big`、`\bigg` 或 `\Bigg`, 这些符号就会被放大:

正常大小	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇕ ⇕
<code>\big</code>	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇕ ⇕
<code>\Big</code>	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇕ ⇕
<code>\bigg</code>	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇕ ⇕
<code>\Bigg</code>	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇕ ⇕

上述命令放大固定的倍数: 1.5 倍、2 倍、2.5 倍、3 倍, 此外还有自适应的放大命令 `\left\cdots\right`, 它们会根据定界符所包围的公式的大小自动变大, 见下例的括号. 输入

```
\[
  \Bigg( \sum_{k=\frac{1}{2}}^{N^2} \Bigg)\quad
  \left( \sum_{k=\frac{1}{2}}^{N^2} \right)
```

输出为

$$\left(\sum_{k=\frac{1}{2}}^{N^2} \right) \quad \left(\sum_{k=\frac{1}{2}}^{N^2} \right)$$

其中即使是放大倍数最大的`\Bigg`也仍显得括号小了点, 而`\left... \right`会随着所围公式的变化而变化。

`\left... \right`总是成对出现的, 缺一不可。此外还需要注意两点: 一是左右两个符号不能排版到两行上, 即中间不能插入换行符; 二是如果只需在公式一侧有自动变化大小的定界符, 那么只要用英文句号(即小圆点)代替另一侧那个不出现的定界符即可, 打印时这个圆点是不会出现的, 例如输入

```
\[
\left.\frac{\partial f(x,y)}{\partial x}\right|_{x=0}
\]
```

输出为

$$\left.\frac{\partial f(x,y)}{\partial x}\right|_{x=0}$$

放大左括号时, 可以将`\big, \dots, \Bigg`等写成`\bigl, \dots, \Biggl`, 放大右括号时写成`\bigr, \dots, \Biggr`, 在 \LaTeX 中它们没有新的作用。但是如果写成`\bigrm, \dots, \Biggrm`, 则被放大的符号就被当成是关系运算符来处理, 即这个符号两侧会插入空白间隔, 对比下例竖线两边的空白。输入

```
\[
a \big| b \quad \quad a \bigrm| b
\]
```

输出为

$$a|b \quad a \big| b$$

§ 4.12 矩阵

使用阵列(array)环境可以把一些元素排列成横竖都对齐的矩形阵列, 命令格式是

```
\begin{array}[竖向位置]{列格式}
  第一行\\
  第二行\\
  .....\\
  第末行
\end{array}
```

其中的参数含义与 `tabular` 表格环境的参数含义完全相同, 每行的输入格式也是相同的, 例如一行中的各列是用 `&` 符号分隔的, 每行都用 `\\` 结束 (最末行不必写换行符), 详见第 37 页及其后的内容. 但这个阵列环境不能用于文字模式, 只能用于数学模式, 换言之, 阵列环境必须位于数学环境之内, 或者用一对 `$` 或一对 `$$` 括起来. 如果是排版矩阵, 左右应使用可以自动调整大小的圆括号或方括号.

下面是一个简单的矩阵例子. 输入

$$\left[\begin{array}{ccc} 11 & 12 & 13 \\ 21 & 22 & 23 \end{array}\right]$$

输出

$$\begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \end{pmatrix}$$

$$f(x) = \begin{cases} x & \text{当 } x \geq 0 \text{ 时;} \\ -x & \text{其他情形.} \end{cases}$$

可如下输入

```
\[
f(x)=\left\{
\begin{array}{ll}
x & \&\mbox{当~$x\ge 0$~时;}\\
-x & \&\mbox{其他情形.}
\end{array}
\right.
```

注意数学公式里出现的汉字必须放在 `\mbox{}` 内, 而在 `\mbox{}` 的内部仍可出现数学模式.

§ 4.13 单行公式与多行公式

单行公式和多行公式显然是指行间公式. 自动编号的单行公式环境是

```
\begin{equation}
  公式
\end{equation}
```

不参与编号的单行公式环境是

```
\begin{displaymath}
  公式
\end{displaymath}
```

可以简写成

```
\[ 公式 \]
```

人工编号的单行公式可使用 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 原有的行间公式标记

```
$$ 公式 \eqno 编号 $$    或    $$ 公式 \leqno 编号 $$
```

上面几处提到的公式既可以是普通的单行公式, 又可以是作为一个整体处理的环境或盒子, 例如是一个 `array` 环境.

如果要改变公式自动编号的值, 需在该公式之前用如下命令设置公式编号计数器的初值:

```
\setcounter{equation}{数}
```

其中的数应是整数. 当该命令位于环境之外时, 其后环境中第一个编号公式的序号值比计数器的值多 1.

为了以后引用自动编号的公式, 可用 `\label{公式标记}` 命令为这个公式加个标记, 以后就用 `\ref{公式标记}` 命令来引用此公式. 不过为了得到正确的编号, 有时还需要用 \LaTeX 再编译一次. 例如输入

```
\setcounter{equation}{5}
\begin{equation}\label{eq:squaresum}
  a^2+b^2=c^2,
\end{equation}
由公式(\ref{eq:squaresum})就可得到所需结论.
```

可得到以下输出

$$a^2 + b^2 = c^2, \quad (6)$$

由公式(6)就可得到所需结论.

多行公式就是包含几行的公式, 所有行都在一个特定位置上下竖直对齐, 这个对齐位置大多是一个关系符号所在位置. 实现这一功能的环境有两个, 一是标准的, 一是带 * 号的:

```
\begin{eqnarray}
  第一行\\
  第二行\\
  .....\\
  第末行
\end{eqnarray}
```

```
\begin{eqnarray*}
  第一行\\
  第二行\\
  .....\\
  第末行
\end{eqnarray*}
```

每行的格式是

```
左边公式 & 中间公式 & 右边公式 \\\
```

也就是说, 每行都有 3 列, 列间用符号 & 分开, 每行用 \\ 结束. 在显示时, 所有行的左边公式在左列中是靠右对齐的, 右边公式在右列中是靠左对齐的, 中间公式在中列中是居中对齐的. 因此每一行就像是环境 `\begin{array}{rcl}...\end{array}` 中的行.

通常中间公式是单个的数学符号, 如关系符号 $=, <, \leq, >, \geq$ 等.

多行公式环境 `eqnarray` 与阵列环境 `array` 是不同的, 有如下区别:

1. `eqnarray` 环境本身就是数学模式, 所以不要再加数学模式标记. 但 `array` 环境是一个只能用于数学模式的表格环境, 它本身并不会自动进入数学模式, 必须用数学模式标记把它括起来.

2. `eqnarray` 环境被排版成行间公式, 而 `array` 环境则根据两边的数学模式标记排版成行内公式或行间公式, 但 `array` 环境内的各列总是按行内公式选取符号的大小尺寸, 例如分数的分子和分母是用角标字体显示的. `eqnarray` 环境的左右两列按行间公式选取符号的尺寸, 但中间一列却是按行内公式选取符号的尺寸, 所以中间一列不要放分数, 一定要放的话, 可显式指定使用 `\displaystyle` 字体.

3. `eqnarray` 环境可以自动对每行公式编号, 而 `array` 环境不能被自动编号, 必要时可显式指定公式的编号.

标准形式 (即不带 * 号) 的 `eqnarray` 环境对每行公式自动编号, 带 * 号的环境则不参与公式编号. 为了对标准形式中的某行不加公式编号, 应在该行换行符 \\ 之前插入命令

`\nonumber`

下面是一个多行公式的例子.

$$d(uv) = (uv)'dx = (u'v + uv')dx \quad (2)$$

$$= v(u'dx) + u(v'dx)$$

$$= vdu + u dv \quad (5)$$

这样就得到了公式(5).

是如下输入的:


```
{
\setlength{\arraycolsep}{2.5pt}
\setcounter{equation}{1}
\begin{eqnarray}
d(uv) &= & (uv)'dx=(u'v+uv') dx\\
&= & v(u'dx)+u(v'dx) \nonumber\\
\setcounter{equation}{5}
&= & v du+u dv \label{leibniz}
\end{eqnarray}
这样就得到了公式(\ref{leibniz}).
}
```

本段文本还示例了命令 `\nonumber` 和 `\setcounter` 的用法和效果. 尤其当为计数器置数的命令位于环境内部时, 其后第一个编号公式的序号值取计数器的值.

在阵列环境中, 参数 `\arraycolsep` 表示每个列的两端留空的长度, 其值是列间距的一半, 默认值比较大, 此处改用了较小的值, 使得两列之间不显得太宽.

下面也是一个多行公式, 有意出现瑕疵:

$$\sum_{i=1}^{20} i = 1+2+3+4+5+6+7+8+9+10 \\ +11+12+13+14+15+16+17 \\ +18+19+20$$

这是如下输入的:

```
\begin{eqnarray*}
\sum_{i=1}^{20} i &= & 1+2+3+4+5+6+7+8+9+10 \\
&& +11+12+13+14+15+16+17 \\
&& {}+18+19+20
\end{eqnarray*}
```

输出的公式很不好看. 首先是行距不同, 这是因为求和号撑大了所在行的总高度. 改进方法是将该行的换行符 `\\` 改成 `\\[-6pt]`, 使下一行向上提升一段距离. 其次是第二行和第三行开始的两个加号既未上下对齐, 又与后面数字的间距不同, 这是由于加号的两个不同意义造成的. 加号作为单目运算符时, 是正负号标志, 与其后的运算量之间间隔很小, 作为双目运算符时, 与两边的运算量之间有稍大的间隔. 输入成 `&&+11` 时, 加号显然是单目运算符, 因为前面的 `&` 是列分隔符, 不是运算量. 输入成 `&&{}+18` 时, 加号就成为双目运算符了, 所以只要将 `&&+11` 改为 `&&{}+11` 就能使二、三两行对齐. 修改后显示为

$$\sum_{i=1}^{20} i = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 \\ + 11 + 12 + 13 + 14 + 15 + 16 + 17 \\ + 18 + 19 + 20$$

§ 4.14 数学字体

4.14.1 选择数学字体

显示变量名时, 默认使用数学斜体. 可以使用的数学字体命令有

```
\mathrm \mathit \mathtt \mathsf \mathbf \mathcal
```

这些命令开始字符都是 `math`, 其后的字母表明了字体形状. 其中最后一个命令 `\mathcal` 是数学花体, 只能用于大写拉丁字母. 使用这些字体命令时, 文本中的空格一律不起作用. 这些字体命令只能用于数学模式中, 例如输入

```
$g=9.8\,\mathrm{m/s^2}$
```

输出是 $g = 9.8 \text{ m/s}^2$. 下面是各种字体的例子:

输入	输出
<code>\mathrm{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathit{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathtt{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathsf{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathbf{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathcal{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$AB\mathcal{X} + -\infty \in \Delta + \sum \sin \alpha$

可见这些命令只影响拉丁字母、数字和大写希腊字母, 不影响数学符号、已定义的函数名称和小写希腊字母. 此外数学花体只应当用于 26 个大写拉丁字母.

利用一些宏包, 还可得到一些特殊字体, 例如:

示例	命令	所需宏包
ABC	<code>\mathcal{ABC}</code>	
\mathcal{ABC}	<code>\mathcal{ABC}</code>	<code>mathrsfs</code>
ABC	<code>\mathcal{ABC}</code>	<code>eucal</code> 带选项 <code>[mathcal]</code> 或者
	<code>\mathscr{ABC}</code>	<code>eucal</code> 带选项 <code>[mathscr]</code>
$\frac{ABCxyz}{3}$	<code>\mathfrak{ABCxyz}</code>	<code>eufrak</code>
ABC	<code>\mathbb{ABC}</code>	<code>amsmaths</code> 或 <code>amssymb</code>

请注意: `\mathcal`, `\mathscr` 和 `\mathbb` 都只能用于大写拉丁字母.

4.14.2 希腊字母

凡是字形与拉丁字母相同的希腊字母, 都可直接从键盘输入. 没有对应符号的, 使用下述命令输入:

小写希腊字母:

α	<code>\alpha</code>	θ	<code>\theta</code>	ξ	<code>\xi</code>	τ	<code>\tau</code>
β	<code>\beta</code>	ϑ	<code>\vartheta</code>	π	<code>\pi</code>	υ	<code>\upsilon</code>
γ	<code>\gamma</code>	ι	<code>\iota</code>	ϖ	<code>\varpi</code>	ϕ	<code>\phi</code>
δ	<code>\delta</code>	κ	<code>\kappa</code>	ρ	<code>\rho</code>	φ	<code>\varphi</code>
ϵ	<code>\epsilon</code>	λ	<code>\lambda</code>	ϱ	<code>\varrho</code>	χ	<code>\chi</code>
ε	<code>\varepsilon</code>	μ	<code>\mu</code>	σ	<code>\sigma</code>	ψ	<code>\psi</code>
ζ	<code>\zeta</code>	ν	<code>\nu</code>	ς	<code>\varsigma</code>	ω	<code>\omega</code>
η	<code>\eta</code>						

大写希腊字母:

Γ	<code>\Gamma</code>	Λ	<code>\Lambda</code>	Σ	<code>\Sigma</code>	Ψ	<code>\Psi</code>
Δ	<code>\Delta</code>	Ξ	<code>\Xi</code>	Υ	<code>\Upsilon</code>	Ω	<code>\Omega</code>
Θ	<code>\Theta</code>	Π	<code>\Pi</code>	Φ	<code>\Phi</code>		

希腊字母只能用于数学模式. 大写希腊字母通常是直立罗马字体, 如若需要斜体的大写希腊字母, 可使用命令 `\mathit` 或 `\mathnormal`, 区别是输出的字符间距不同, 例如

`$|\mathnormal{\Gamma\Theta\Xi\Omega\Pi}|$` 输出 $|\Gamma\Theta\Xi\Omega\Pi|$.

`$|\mathit{\Gamma\Theta\Xi\Omega\Pi}|$` 输出 $|\Gamma\Theta\Xi\Omega\Pi|$.

对于拉丁字母, `\mathnormal` 与 `\mathit` 也有同样的区别, 即后者字符的宽度和间距都稍紧一些.

在使用 ISO 国际标准时会遇到打印直立的小写希腊字母的问题. 一种方法是在导言区加入调入宏包 `pxfonts` 的命令 `\usepackage{pxfonts}`, 在数学模式里使用以下命令就可得到直立的小写希腊字母:

α	<code>\alphaup</code>	θ	<code>\thetaup</code>	ξ	<code>\xiup</code>	τ	<code>\tauup</code>
β	<code>\betaup</code>	ϑ	<code>\varthetaup</code>	π	<code>\piup</code>	υ	<code>\upsilonup</code>
γ	<code>\gammaup</code>	ι	<code>\iotaup</code>	ϖ	<code>\varpiup</code>	ϕ	<code>\phiup</code>
δ	<code>\deltaup</code>	κ	<code>\kappaup</code>	ρ	<code>\rhoup</code>	φ	<code>\varphiup</code>
ϵ	<code>\epsilonup</code>	λ	<code>\lambdaup</code>	ϱ	<code>\varrhoup</code>	χ	<code>\chiup</code>
ε	<code>\varepsilonup</code>	μ	<code>\muup</code>	σ	<code>\sigmaup</code>	ψ	<code>\psiup</code>
ζ	<code>\zetaup</code>	ν	<code>\nuup</code>	ς	<code>\varsigmaup</code>	ω	<code>\omegaup</code>
η	<code>\etaup</code>						

不过调入宏包 `pxfonts` 后会引引起整个字样的改变, 代价太大. 为此笔者从这个宏包中提取出有关直立小写希腊字母的定义命令, 做成一个格式文件 `greekup.sty`, 读者只需在导言区加入命令 `\usepackage{greekup}`, 调用此宏包, 同样可以使用上列命令得到直立的小写希腊字母. 参见 4-14-1.tex. 文件 `greekup.sty` 放在随书

光盘的 Doc\本书例题 子目录里, 欲使用的读者可将其复制入 L^AT_EX 源文件所在的目录里, 或者复制入 localtexmf\tex\latex 的任何一个子目录里. 不过别忘了更新 File Name database (在 WinEdt 里点击下拉菜单 Accessories→MiKTeX Options, 在弹出窗口里点击按钮“Refresh Now”即可), 使 L^AT_EX 能找到它.

如果你只需要使用代表圆周率的 π , 那么只要在导言区加入以下两行命令就可以了 (见例 4-14-2.tex):

```
\DeclareSymbolFont{lettersA}{U}{pxmia}{m}{it}
\DeclareMathSymbol{\piup}{\mathord}{lettersA}{"19}
```

4.14.3 数学粗体

为了将公式中的所有符号都变成粗体, 可使用命令

```
\boldmath \unboldmath
```

前者把数学公式中的大写和小写拉丁字母、小写希腊字母 和其他符号设成粗体 (但仍有一些符号例外, 例如具有两种尺寸的符号), 后者取消前者的作用, 把数学字体重新设成正常字体.

注意这两个命令虽然是数学字体命令, 但不能用于数学模式中. 一种用法是在进入数学模式之前使用命令 `\boldmath`, 退出数学模式之后立即使用 `\unboldmath`. 另一种用法是在数学模式内部使用 LR-盒子 `\mbox{\boldmath$...$}`, 将部分公式设成粗体.

如果在导言区使用命令

```
\usepackage{bm}
```

加载宏包, 就可以使用命令

```
\boldsymbol{数学符号} 或 \bm{数学符号}
```

将数学符号 (包括具有两种尺寸的符号) 设成粗体. 如果对整个公式使用这个命令, 那么整个公式都是粗体了. 可以说只记命令 `\bm`, 不必再记其他的数学粗体命令了. 这种粗体对具有两种尺寸的符号是通过微小平移多次打印同一对象得到的. 最后给出一个例子, 输入

```
\[
\Delta = \sum_{k=1}^n \sin \delta_{k} + A_k \sim
\bm{\Delta = \sum_{k=1}^n \sin \delta_{k} + A_k}
\]
```

输出为

$$\Delta = \sum_{k=1}^n \sin \delta_k + A_k \sim \Delta = \sum_{k=1}^n \sin \delta_k + A_k$$

注意: 若在公式中使用直立粗体拉丁字母, 仍需使用 `\mathbf` 命令.
使用 `amsmath` 也可打印数学粗体公式, 参见第 142 页.

§ 4.15 数学符号表

绝大多数的数学符号不能直接从键盘上得到, 只能使用命令序列产生各种符号. 去掉命令的前缀倒斜线, 剩下的字符串基本反映了符号的数学名称. 数学符号多种多样, 前面介绍过的希腊字母和数学重音字母也可看成是数学符号. 下面分类列出数学符号和对应的命令, 其中标有下画线的命令表示需要事先加载宏包 `latexsym`, 当然命令本身是没有下画线的. 此外, 对个别符号给出了两种命令.

除了具有两种尺寸的符号之外, 数学模式内的其他符号和字符都受当前字体尺寸的控制, 但是不要将选择字体尺寸的命令放在数学模式内部, 否则这种命令不起任何作用.

4.15.1 二元运算符

\pm	<code>\pm</code>	\cap	<code>\cap</code>	\circ	<code>\circ</code>	\bigcirc	<code>\bigcirc</code>
\mp	<code>\mp</code>	\cup	<code>\cup</code>	\bullet	<code>\bullet</code>	\square	<code>\Box</code>
\times	<code>\times</code>	\uplus	<code>\uplus</code>	\diamond	<code>\diamond</code>	\Diamond	<code>\Diamond</code>
\div	<code>\div</code>	\sqcap	<code>\sqcap</code>	\triangleleft	<code>\lhd</code>	\bigtriangleup	<code>\bigtriangleup</code>
\cdot	<code>\cdot</code>	\sqcup	<code>\sqcup</code>	\triangleright	<code>\rhd</code>	\bigtriangledown	<code>\bigtriangledown</code>
$*$	<code>\ast</code>	\vee	<code>\vee</code>	\trianglelefteq	<code>\unlhd</code>	\triangleleft	<code>\triangleleft</code>
\star	<code>\star</code>	\wedge	<code>\wedge</code>	\trianglerighteq	<code>\unrhd</code>	\triangleright	<code>\triangleright</code>
\dagger	<code>\dagger</code>	\oplus	<code>\oplus</code>	\oslash	<code>\oslash</code>	\setminus	<code>\setminus</code>
\ddagger	<code>\ddagger</code>	\ominus	<code>\ominus</code>	\odot	<code>\odot</code>	\wr	<code>\wr</code>
\amalg	<code>\amalg</code>	\otimes	<code>\otimes</code>				

4.15.2 关系运算符

\leq	<code>\le</code> 或 <code>\leq</code>	\geq	<code>\ge</code> 或 <code>\geq</code>	\neq	<code>\ne</code> 或 <code>\neq</code>	\sim	<code>\sim</code>
\ll	<code>\ll</code>	\gg	<code>\gg</code>	\doteq	<code>\doteq</code>	\simeq	<code>\simeq</code>
\subset	<code>\subset</code>	\supset	<code>\supset</code>	\approx	<code>\approx</code>	\asymp	<code>\asymp</code>
\subseteq	<code>\subseteq</code>	\supseteq	<code>\supseteq</code>	\cong	<code>\cong</code>	\smile	<code>\smile</code>
\sqsubset	<code>\sqsubset</code>	\sqsupset	<code>\sqsupset</code>	\equiv	<code>\equiv</code>	\frown	<code>\frown</code>
\sqsubseteq	<code>\sqsubseteq</code>	\sqsupseteq	<code>\sqsupseteq</code>	\propto	<code>\propto</code>	\bowtie	<code>\bowtie</code>
\in	<code>\in</code>	\ni	<code>\ni</code>	\prec	<code>\prec</code>	\succ	<code>\succ</code>
\vdash	<code>\vdash</code>	\dashv	<code>\dashv</code>	\preceq	<code>\preceq</code>	\succeq	<code>\succeq</code>

\models	<code>\models</code>	\perp	<code>\perp</code>	\parallel	<code>\parallel</code>	$ $	<code>\mid</code>
$=$	<code>=</code>	$>$	<code>></code>	$<$	<code><</code>		

最后 3 个符号可以直接从键盘上输入, 没有对应的命令, 写在此处仅仅是为了强调它们也是关系运算符。

符号 \parallel 与 $|$ 作为定界符出现过, 见第 55 页。但现在作为关系运算符出现的, 与作为定界符时的命令是不同的。

关系运算符与两侧的变量之间有一些额外的空白, 而定界符是不带额外空白的, 例如输入 `$a\mid b$` 和 `$a|b$`, 输出结果为 $a|b$ 和 $a|b$, 显然是不同的, 使用时不要混淆。如果要取消关系运算符两侧的额外空白, 只需简单地用花括号把它括起了。例如输入 `$a=b$, $a{=}b$, $a{=b}$, $a{=}b$`, 输出是 $a=b$, $a=b$, $a=b$, $a=b$, 紧靠着花括号那一侧的额外空白不见了。

在关系运算符上画一个斜线就是否定形式的关系运算符, 例如在“ \equiv ”上画一斜线得到“ \ncong ”。在一个符号上画斜线的“通用”方法是在其命令前面加上 `\not`。对于不等号, 由于经常使用, 除了可由 `\not=` 产生外, 还可以使用上面表中已定义的命令 `\ne` 或 `\neq`。下表是一些否定形式的关系运算符:

\nless	<code>\not<</code>	\ngtr	<code>\not></code>	\neq	<code>\not=</code>
\nleq	<code>\not\leq</code>	\ngeq	<code>\not\geq</code>	\ncong	<code>\not\equiv</code>
\nprec	<code>\not\prec</code>	\nsucc	<code>\not\succ</code>	\nsim	<code>\not\sim</code>
\npreceq	<code>\not\preceq</code>	\nsucceq	<code>\not\succeq</code>	\nsimeq	<code>\not\simeq</code>
\nsubset	<code>\not\subset</code>	\nsupset	<code>\not\supset</code>	\napprox	<code>\not\approx</code>
\nsubseteq	<code>\not\subseteq</code>	\nsupseteq	<code>\not\supseteq</code>	\ncong	<code>\not\cong</code>
\nsubsetneq	<code>\not\subsetneq</code>	\nsupsetneq	<code>\not\supsetneq</code>	\nasymp	<code>\not\asymp</code>
\ni	<code>\not\ni</code>	\nin	<code>\not\in</code>	\notin	<code>\notin</code>

要特别注意最后两个符号, 在数学上这实际是同一个符号, 只不过是用不同命令生成的, 显然专用命令 `\notin` 生成的符号 \notin 优于通用命令 `\not\in` 生成的符号 \notin 。

4.15.3 箭头符号

\leftarrow	<code>\leftarrow</code> 或 <code>\gets</code>	\longleftarrow	<code>\longleftarrow</code>	\uparrow	<code>\uparrow</code>
\Leftarrow	<code>\Leftarrow</code>	\Longleftarrow	<code>\Longleftarrow</code>	\Uparrow	<code>\Uparrow</code>
\rightarrow	<code>\rightarrow</code> 或 <code>\to</code>	\longrightarrow	<code>\longrightarrow</code>	\downarrow	<code>\downarrow</code>
\Rightarrow	<code>\Rightarrow</code>	\Longrightarrow	<code>\Longrightarrow</code>	\Downarrow	<code>\Downarrow</code>
\leftrightarrow	<code>\leftrightarrow</code>	\longleftrightarrow	<code>\longleftrightarrow</code>	\Updownarrow	<code>\Updownarrow</code>
\Leftrightarrow	<code>\Leftrightarrow</code>	\Longleftrightarrow	<code>\Longleftrightarrow</code>	\Updownarrow	<code>\Updownarrow</code>
\mapsto	<code>\mapsto</code>	\longmapsto	<code>\longmapsto</code>	\nearrow	<code>\nearrow</code>
\hookrightarrow	<code>\hookrightarrow</code>	\hookrightarrow	<code>\hookrightarrow</code>	\searrow	<code>\searrow</code>
\leftharpoonup	<code>\leftharpoonup</code>	\rightharpoonup	<code>\rightharpoonup</code>	\swarrow	<code>\swarrow</code>
\leftharpoondown	<code>\leftharpoondown</code>	\rightharpoondown	<code>\rightharpoondown</code>	\nwarrow	<code>\nwarrow</code>
\Rightarrow	<code>\Rightarrow</code>	\leadsto	<code>\leadsto</code>	\iff	<code>\iff</code>

表中的`\Longlefttrightarrow`生成的箭头(\Longleftrightarrow)是普通箭头,而`\iff`生成的是数学中表示等价关系的箭头(\iff),后者的两侧有额外的空白,实际是一个关系运算符。

4.15.4 其他符号

为了使相关符号放在一起,下表中的个别符号与前面表格有少许重复。

\aleph	<code>\aleph</code>	$'$	<code>\prime</code>	\forall	<code>\forall</code>	\square	<code>\Box</code>
\hbar	<code>\hbar</code>	\emptyset	<code>\emptyset</code>	\exists	<code>\exists</code>	\diamond	<code>\Diamond</code>
\imath	<code>\imath</code>	∇	<code>\nabla</code>	\neg	<code>\neg</code>	\triangle	<code>\triangle</code>
\jmath	<code>\jmath</code>	$\sqrt{}$	<code>\surd</code>	\flat	<code>\flat</code>	\clubsuit	<code>\clubsuit</code>
ℓ	<code>\ell</code>	∂	<code>\partial</code>	\natural	<code>\natural</code>	\diamondsuit	<code>\diamondsuit</code>
\wp	<code>\wp</code>	\top	<code>\top</code>	\sharp	<code>\sharp</code>	\heartsuit	<code>\heartsuit</code>
\Re	<code>\Re</code>	\bot	<code>\bot</code>	\parallel	<code>\parallel</code>	\spadesuit	<code>\spadesuit</code>
\Im	<code>\Im</code>	\vdash	<code>\vdash</code>	\angle	<code>\angle</code>	\Join	<code>\Join</code>
\mho	<code>\mho</code>	\dashv	<code>\dashv</code>	\backslash	<code>\backslash</code>	∞	<code>\infty</code>

4.15.5 具有两种尺寸的符号

有些符号具有两种尺寸,位于行内公式时,显示为小尺寸,位于行间公式时,显示为大尺寸,下表列出了同一命令表示的不同大小的符号。

\sum	\sum	<code>\sum</code>	\cap	\cap	<code>\bigcap</code>	\odot	\odot	<code>\bigodot</code>
\int	\int	<code>\int</code>	\cup	\cup	<code>\bigcup</code>	\otimes	\otimes	<code>\bigotimes</code>
\oint	\oint	<code>\oint</code>	\sqcup	\sqcup	<code>\bigsqcup</code>	\oplus	\oplus	<code>\bigoplus</code>
\prod	\prod	<code>\prod</code>	\vee	\vee	<code>\bigvee</code>	\uplus	\uplus	<code>\biguplus</code>
\coprod	\coprod	<code>\coprod</code>	\wedge	\wedge	<code>\bigwedge</code>			

上述符号都可以使用上下标命令加上下限,上下限的位置在行内公式和行间公式中可能有所不同.总可以使用命令`\limits`强制上下限放在符号的上方和下方,或者使用命令`\nolimits`强制上下限放在符号的侧旁.参见第50页的例子。

4.15.6 函数名

在数学公式中显示变量名和函数名时有一个通行的标准,就是用斜体显示变量名,用正体显示函数名.在 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 中通过在函数名前加上倒斜线,将函数名与变量名区分开.这样的函数名需要事先定义, $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 已定义了下列函数名:

<code>\arccos</code>	<code>\cosh</code>	<code>\det</code>	<code>\inf</code>	<code>\limsup</code>	<code>\pr</code>	<code>\tan</code>
<code>\arcsin</code>	<code>\cot</code>	<code>\dim</code>	<code>\ker</code>	<code>\ln</code>	<code>\sec</code>	<code>\tanh</code>
<code>\arctan</code>	<code>\coth</code>	<code>\exp</code>	<code>\lg</code>	<code>\log</code>	<code>\sin</code>	<code>\bmod</code>

<code>\arg</code>	<code>\csc</code>	<code>\gcd</code>	<code>\lim</code>	<code>\max</code>	<code>\sinh</code>	<code>\pmod</code>
<code>\cos</code>	<code>\deg</code>	<code>\hom</code>	<code>\liminf</code>	<code>\min</code>	<code>\sup</code>	

最右列下方两个命令都生成函数 mod, 所以把它两个排在一起, 未按字母次序排在表中. 这两个函数用法不同, 输出形式也不同, 示例如下:

输入 `$a\bmod(m+n)$`, 输出为 $a \bmod (m+n)$.

输入 `$a\pmod{m+n}$`, 输出为 $a \pmod{m+n}$.

第五章 常用文档的类别与版式

在 \LaTeX 源文件导言部分的开头有一条文档类别命令,确定整篇文档的全局处理格式,命令形如

`\documentclass[可选项]{文档类别}`

其中常用的标准文档类别有4种:

<code>book</code>	书籍类
<code>report</code>	报告类
<code>article</code>	文章类
<code>letter</code>	书信类

必须从中选用一种而且只能选用一种. 不同的文档类别具有不同的默认版式和不同的成分,例如`book`类,允许含有`chapter`(章),并对奇偶页有不同的处理方式,而`article`类就没有“章”这一层次. 一般我们书写数学论文往往选用“`article`”类.

学完本章,读者就可以尝试打印自己的论文了. 如果在排版中发现问题或者想使自己论文的版面更漂亮些,就需要参看“提高篇”了.

§ 5.1 文档类别命令中的可选项

文档类别命令中可选项的作用范围是整篇文档,本章先介绍一些最常用的可选项. 注意所有选项都不能加前导符“ \backslash ”. 如果使用了多个互不排斥的选项,这些选项应用逗号分开(只能用西文逗号,不可用中文逗号).

5.1.1 指定基本字体尺寸

指定基本字体尺寸的选项有3种:

`10pt` `11pt` `12pt`

只能选用一种. 若都不选用,则默认是选用`10pt`. 除非在文稿中加入了另外的声明,否则所有普通文本都使用这个指定的字体尺寸,其他变大变小的字体都是相对于这个尺寸的,各种标题、角标、脚注等都会随着这个指定尺寸的改变而自动改变大小.

5.1.2 指定纸张大小

如果不想直接指定页芯的大小(见第 11 页), 可改为指定纸张的大小, \LaTeX 会根据指定的字体尺寸和纸张大小自动确定页芯的大小. 指定纸张大小的选项有如下几种, 括号中的数字是纸张尺寸的说明, 不要输入到源文件中.

<code>a4paper</code> (297×210mm)	<code>executivepaper</code> (10.5×7.25in)
<code>a5paper</code> (210×148mm)	<code>legalpaper</code> (14×8.5in)
<code>b5paper</code> (250×176mm)	<code>letterpaper</code> (11×8.5in)

默认值是 `letterpaper`, 即美国信纸尺寸.

通常情况下排版输出使用纵向模式 (`portrait` — “肖像画”模式), 即纸张格式中长的方向是排版结果的竖直方向. 如果想使用横向模式, 即将纸张格式中短的方向变为排版输出结果的竖直方向, 可使用如下的“风景画”模式选项:

<code>landscape</code>

注意打印时纸张的摆放方向并不改变.

5.1.3 其他一些选项

与标题页有关的选项是

<code>notitlepage</code>	<code>titlepage</code>
--------------------------	------------------------

对于 `book` 和 `report` 类, 标题默认打印在单独一页. `notitlepage` 选项使标题与开头的文本排到同一页上.

对于 `article` 类, 默认标题与正文排在同一页. `titlepage` 选项可使标题单独成页.

下面又是一对有相反作用的选项

<code>draft</code>	<code>final</code>
--------------------	--------------------

使用选项 `draft` 后, 若排版结果出现超长的行, 则在该行的右端显示一个粗黑条, 提醒用户注意. 默认值是 `final`, 无论一行超出多少, 也不显示粗黑条. 写初稿时先使用选项 `draft`, 等一切调整妥当后再改为 `final`.

§ 5.2 章节

在 \LaTeX 中有一些划分章节的命令, 其一般格式是

<code>\章节命令[短标题]{标题}</code>
<code>\章节命令*{标题}</code>

可用的章节命令按从大到小的层次有如下几种

<code>\part</code>	<code>\chapter</code>	<code>\section</code>	<code>\subsection</code>
<code>\subsubsection</code>	<code>\paragraph</code>	<code>\subparagraph</code>	

命令中的短标题用于显示在目录和 headings 页面版式的页眉中, 如果不选用短标题, 就认为它和标题是一样的.

对于每个不带 * 号的章节命令, 都有一个内部计数器, 计数器的名称与去掉倒斜线的章节命令相同. 每调用一次不带 * 号的章节命令, 对应计数器的值就加 1, 并同时将低层次的计数器重置为 0.

不带 * 号的章节命令会自动对章节进行顺序编号, 并可自动形成目录表和页眉标题 (这些也是 L^AT_EX 的特色). 带有 * 号的章节命令则不会自动编号, 也不会把章节标题放到目录表和页眉中.

每个章节命令都有一个层次号, `\section` 层次号总是 1, 向下逐次递增, 直到 `\subparagraph` 层次号为 5. 对于 book 和 report 类, `\part` 层次号为 -1, `\chapter` 层次号为 0. 对于 article 类, `\part` 层次号为 0, 没有 `\chapter` 这一层次.

自动编号时, 除了对 `\part` 独立编号外, 还对其后的前 3 个层次进行关联编号: 对于 book 和 report 类, 就是对章 (`\chapter`)、节 (`\section`) 和小节 (`\subsection`) 进行编号, 即编号到层次号为 2 的层次, 其中章有独立编号, 节的编号前缀有章的编号, 小节的编号前缀有章和节的编号; 对于 article 类, 进行编号的 3 个层次是 `\section`、`\subsection` 和 `\subsubsection`, 即编号到层次号为 3 的层次, 其中节的编号是独立的, 小节的编号前面缀有节的编号, 次小节的编号前面缀有节和小节的编号. 计数器 `secnumdepth` 用于记录编号到达的最深层次数, 由上可知, 对于 book 和 report 类, 其值为 2, 对于 article 类, 其值为 3.

命令

<code>\setcounter{secnumdepth}{数}</code>
--

用于扩展或减小章节编号的层次. 对于 book 和 report 类, 数的取值范围是 -2 到 5; 对于 article 类, 数的取值范围是 -1 到 5. 细心的读者会发现数的范围中出现了不存在的层次数, 使用这个不存在的层次数意味着禁止所有的章节编号.

L^AT_EX 内部有一批计数器, 储存当前的章节号, 它们的名称就是相应的章节命令除去前缀符 “\”, 它们分别是: `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph`, `subparagraph`. 如果想改变章节计数器的初始值, 可用如下形式的命令

<code>\setcounter{章节计数器的名称}{数}</code>

这个命令应放在章节命令之前, 新的章节的序号值是这个初始值加 1.

`\part` 标题比较特殊, 它被分成两行, 第一行输出 “Part I”, 字体为 “`\Large \bfseries`”, 第二行再输出具体的标题. 如果要使第一行符合中文习惯, 并且只作最少的改动, 可在 CJK 环境内使用以下两条命令:

```
\renewcommand{\partname}{}\n\renewcommand{\thepart}{第\, \Roman{part}\, 篇}
```

下面的例 5-2-1.tex 展示了在 article 的默认格式下章节标题的样式以及它们与正文之间的留空情况。

```
% 5-2-1.tex\n\documentclass{article}\n\usepackage{CJK}\n\begin{document}\n\begin{CJK*}{GBK}{song}\n\CJKtilde\n\renewcommand{\partname}{}\n\renewcommand{\thepart}{第\, \Roman{part}\, 篇}\n\part{篇 (Part) 的标题}\n本篇将介绍基本概念与结论.\n\setcounter{section}{3}\n\section{节 (Section) 的标题}\n本节包括以下内容.\n\subsection{小节 (Subsection) 的标题}\n本小节将介绍以下基本概念与结论.\n\subsubsection{子节 (Subsubsection) 的标题}\n本子节将介绍以下基本概念与结论.\n\paragraph{段 (Paragraph) 的标题}\n本段将介绍以下基本概念与结论.\n\subparagraph{子段 (Subparagraph) 的标题}\n本子段将介绍以下基本概念与结论.\n\end{CJK*}\n\end{document}
```

其输出为

第 I 篇

篇(Part)的标题

本篇将介绍基本概念与结论.

4 节(Section)的标题

本节包括以下内容.

4.1 小节(Subsection)的标题

本小节将介绍以下基本概念与结论.

4.1.1 子节(Subsubsection)的标题

本子节将介绍以下基本概念与结论.

段(Paragraph)的标题 本段将介绍以下基本概念与结论.

子段(Subparagraph)的标题 本子段将介绍以下基本概念与结论.

§ 5.3 文章标题

在 L^AT_EX 中定义了几个有关文章标题的命令:

```
\title{标题文本}
\author{作者信息}
\date{日期文本}
\thanks{脚注文本}
\maketitle
```

L^AT_EX 用特定的字体字号居中打印标题内容, 如果标题过长, 会自动分行. 作者也可用命令 `\\` 人工分行, 此时命令 (以 `\title` 为例) 形如

```
\title{...\\...\\...}
```

如果有多个作者, 其作者信息可以用 `\and` 相互隔开, 如

```
\author 作者1\\工作单位1\\地址1
\and 作者2\\工作单位2\\地址2
```

排版后形如

作者1	作者2
工作单位1	工作单位2
地址1	地址2

如果不想左右并排而是上下排列, 只需用`\\[距离]`代替`\and`.

如果没有给出`\date`命令, 系统会在作者下面自动加上当前日期. 如果给出了`\date`命令, 则不再打印当前日期, 而是在打印日期的地方打印日期文本, 但这个日期文本不一定是日期, 可以是任何文本. 命令`\date{}`相当于取消了`\date`的作用.

命令`\thanks`可以出现在`\title`、`\author`和`\date`命令参数中的任何地方. 它在出现的位置自动加上一个标记, 同时在标题页上以带有相同标记的脚注形式排版脚注文本.

上述命令仅仅给出了与标题有关的信息, 要想真正打印出这些信息, 必须使用命令`\maketitle`, 该命令在它出现的位置开始新的一页打印标题页. 因为文章标题通常要打印在最前面, 所以这个命令通常是正文主体的第一条命令, 即紧接在`\begin{document}`之后.

对于`book`和`report`类, 标题页是单独一页, 只含有`\title`、`\author`和`\date`命令提供的标题信息. 对于`article`类, 标题页上除了标题信息外, 下面还有正文. 文档类别命令中的选项`titlepage`和`notitlepage`可以改变上述默认设置.

如果不想使用上述几个标题命令生成格式化的标题, 而想随心所欲排版标题页, 则应使用下述环境:

```
\begin{titlepage}
  标题页内容
\end{titlepage}
```

其中标题页内容可以含有排版命令和文本, 因此可以排成任何想要的形式. 这个环境结束时, 在新的一页显示排好了的标题页, 所以这个环境不需要`\maketitle`命令. 对于`article`类, 也可以利用这个环境打印单独的标题页.

注意: 上述环境总是重置页码计数器, 从而使标题页的页码是1(但不显示这个页码), 后继页码从2开始. 为了保持页码的连续性, 上述环境最好是放在正文的开始部分.

§ 5.4 摘要

生成摘要的环境是

```
\begin{abstract}
  摘要文本
\end{abstract}
```

在 book 类中没有这种摘要; 在 report 类中, 摘要位于单独一页并带有页码; 在 article 类, 摘要与文章标题打印在同一页, 但若选用了文档类选项 titlepage, 摘要也是单独占一页的。

不过摘要环境自动生成的标题是英文 “Abstract”, 为了生成中文标题, 我们可在使用此环境前插入以下命令:

```
\renewcommand{\abstractname}{摘要}
```

下面的例子 5-4-1.tex 展示了论文首页的输入方法, 书写数学论文时可以照抄其中的控制命令。

```
% 5-4-1.tex
\documentclass[a4paper,12pt]{article}
% [...]为可选项
\usepackage[indentfirst,latexsym,bm]
\usepackage{CJK}
\setlength{\parindent}{2em} % 设置段首缩进量
\renewcommand{\baselinestretch}{1.2} % 重设行距
\begin{document}
\begin{CJK*}{GBK}{song} \CJKtilde
\title{广义特征值问题~$Ax=\lambda Bx$}
\thanks{Supported in part by NSFC.}
\footnotetext{\textit{Keyword}: 广义特征值, ...}
} % 标题、关键字等
\author{Zhang Aiguo (张爱国)\\[5pt]
Department of Mathematics, ECNU\\
(华东师范大学数学系)
} % 作者、工作单位
\date{2001年4月1日}
% {}内可写任意字符或为空白, 无此句则打印当前时间
\maketitle % 打印 \title、\author、\date 等内容
\renewcommand{\abstractname}{摘要}
\begin{abstract}
An abstract {摘要内容}
\end{abstract}
... 正文

\newpage
\end{CJK*}
\end{document}
```

文件最后写上 \newpage 是避免当前 CJK 宏包一个内部的小问题而采取的权宜措施。

相应的输出为

广义特征值问题 $Ax = \lambda Bx^*$

Zhang Aiguo (张爱国)

Department of Mathematics, ECNU

(华东师范大学数学系)

2001年4月1日

摘 要

An abstract 摘要内容

... 正文

**Keyword*: 广义特征值, ...

*Supported in part by NSFC.

第六章 图形

L^AT_EX 提供了若干绘图命令,可以用来绘制直线、矢量线、圆、矩形、圆角矩形(卵形线)、Bézier 曲线等基本图形.在 L^AT_EX 源文件中还可以插入一些外部图形.用 L^AT_EX 命令画图时,对线段的倾角和圆的直径都有一定的限制,如果使用天元软件内嵌的绘图功能,则可画任意斜率的线段和任意半径的圆,并且还有更强的绘图功能,例如只要给出曲线的方程,就能画出二维或三维曲线的图形.

§ 6.1 图形与坐标系

精确绘图离不开坐标系,在 L^AT_EX 中使用的是右手直角坐标系,水平轴为 x 轴,向右为正,竖直轴为 y 轴,向上为正.在两个坐标轴上设置同样大小的单位长度(`\unitlength`),习惯上令单位长度为 1mm,用其他的值也可以,例如令单位长度为 0.8mm.图形中的所有长度都是以单位长度作为度量单位.默认值是 1pt,重置单位长度的命令是

```
\setlength{\unitlength}{长度}
```

当一个图画好后,只要重设单位长度,就相当于对图形作放大或缩小变换.

绘制的任何一个图形总是和一个坐标系固连在一起,坐标轴的原点可以放在任何地方,通常是放在图形区域的左下角.

在 L^AT_EX 中绘制图形需要使用绘图环境:

```
\begin{picture}(宽度,高度)(x 坐标,y 坐标)
    各种绘图命令
\end{picture}
```

环境中的(宽度,高度)是图形占据的区域的宽度和高度,更确切地说,是 L^AT_EX 为插入图形所保留的区域的宽度和高度,换言之,如果实际绘制的图形大于这个尺寸,就可能与其它文本重叠,如果实际图形远远小于这个区域,排版出来就会出现很大的空白,所以应尽量使这个宽度和高度是实际图形的大小.

T_EX 把一切对象都看成是盒子,图形也是一个盒子,这个盒子的宽度和高度就是绘图环境中指定的(宽度,高度),盒子的深度是 0,盒子的基准点(参考点)就是盒子的左下角,L^AT_EX 处理这个图形盒子就像是处理一个的特大的字符盒子.

绘制的任何图形都固连着一个坐标系. 绘图环境中的(x 坐标, y 坐标)是图形盒子的基准点在这个坐标系中的坐标, 实质上是确定了坐标系在盒子中的位置, 也就是确定了整个图形在盒子中的位置. 如果在绘图环境中省略了这个坐标, 则默认这个坐标为 $(0,0)$. 当图形绘制完成后, 如果改变了这个坐标, 就相当于将坐标系在盒子中作了平移. 因为坐标系是固连在图形上的, 所以也就是将图形在图形盒子中作了平移.

确定图形元素在坐标系中位置的命令 (定位命令) 有两个:

```
\put(x, y){图形元素}
\multiput(x, y)(\Delta x, \Delta y){个数}{图形元素}
```

其中图形元素可以是字符串或是下一节中要讲的能够产生图形元素的绘图命令. (x, y) 是图形元素基准点的坐标. 命令 `\multiput` 产生指定个数的图形元素, 这是通过把同样的图形元素进行平移产生的, 平移向量是 $(\Delta x, \Delta y)$. 例如

```
\multiput(3.0,4.0)(1.2,2.5){3}{图形元素}
```

就是在 3 个位置 $(3.0, 4.0)$, $(4.2, 6.5)$, $(5.4, 9.0)$ 画同样的图形元素.

§ 6.2 基本绘图命令

6.2.1 直线和矢量线

使用 L^AT_EX 的绘图环境可以绘制任意长度的水平直线段 (横线)、竖直直线段 (竖线) 和某些特定斜率的斜线段, 命令为

```
\line(\Delta x, \Delta y){长度}
```

对于横线和竖线, 长度是它们的实际长度 (以 `\unitlength` 的值作单位), 对于其他倾斜的直线段, 这个长度是它们在横轴上的投影的长度. 直线段的起点由定位命令 `\put` 或 `\multiput` 给出的坐标 (x, y) 所确定, 直线的方向数是 $(\Delta x, \Delta y)$. 没学过直线方向数的读者可如下理解方向数: 从直线段上的起点开始, 沿 x 轴方向移动 Δx 距离, 再沿 y 轴方向移动 Δy 距离, 就又回到了直线段或它的延长线上. 对方向数有如下限制:

- 必须是整数;
- 取值只能是 $0, \pm 1, \pm 2, \pm 3, \pm 4, \pm 5, \pm 6$;
- Δx 与 Δy 不能有公因子.

若 Δx 与 Δy 有公因子, 编译时会出现出乎意料的错误, 如超出内存.

对于斜线, 长度的值不能小于 10pt 或 3.5mm, 否则不会显示出来, 对于横线和竖线无此限制. 为方便, 以下将直线段称之为直线.

绘制矢量线 (即带有箭头的直线) 命令是:

`\vector(Δx , Δy){长度}`

参数的含义与 `\line` 命令的参数一样, 与直线的区别只是在终点画上了箭头.

对于矢量线的方向数有更严格的限制, 除了必须是整数且无公因子外, 取值只能是 $0, \pm 1, \pm 2, \pm 3, \pm 4$;

当矢量线不是水平或竖直时, 对长度的值的限制与直线一样, 不能小于 10pt 或 3.5mm, 否则显示出来只有箭头部分而没有直线部分.

指定直线或曲线粗细的声明是细线声明或粗线声明:

`\thinlines` 或 `\thicklines`

默认是细线声明 `\thinlines`. 声明的作用直到遇到相反的声明为止, 或遇到所在环境的结束.

对于横线和竖线, 包括水平和竖直的矢量线的直线部分以及用横线和竖线构成的方框, 可用下列声明指定线的粗细:

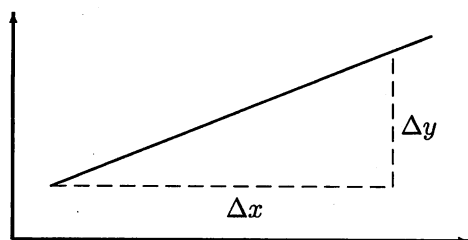
`\linethickness{粗细}`

参数值粗细是正的长度, 必须带有单位, 例如 `{2pt}`, 它与 `\unitlength` (单位长度) 无关. 这个声明遇到细线声明或粗线声明之后就失去作用, 且只对横线和竖线起作用. 对于水平或竖直的矢量线, 当直线部分过粗时, 将会遮住箭头.

下面是一个例子 (见 6-2-1.tex), 输入

```
\begin{center}
\setlength{\unitlength}{1mm}
\begin{picture}(60,30)
\linethickness{1pt}
\put(0,0){\vector(1,0){60}}
\put(0,0){\vector(0,1){30}}
\thicklines
\put(5,7){\line(5,2){50}}
\thinlines
\multiput(5,7)(3,0){15}{\line(1,0){2}}
\multiput(50,7.00)(0,3){6}{\line(0,1){2}}
\put(28,3){$\Delta x$}
\put(51,14){$\Delta y$}
\end{picture}
\end{center}
```

输出为



除非特别说明,以后所有例子均使用1mm作单位长度.

6.2.2 圆和圆角矩形

画圆的命令是

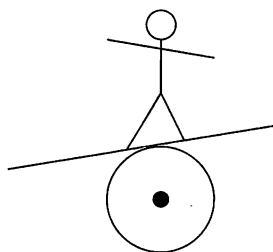
```
\circle{直径}
\circle*{直径}
```

其中第一个命令只画出圆周,第二个命令(带*号的命令)画实心圆.圆周是由若干段小圆弧组成.由于事先设计好的小圆弧只有有限多种,并不具有任意曲率,所以只能画特定尺寸的圆,LaTeX会选出与指定直径最接近的圆.在LaTeX中圆的直径不能超过40pt(约14mm),实心圆直径不能超过15pt(约5.3mm).有些软件不用小圆弧拼成圆,就没有上述限制了.

画圆时定位命令\put或\multiput中的坐标是圆心的坐标.

下面是一个例子(见6-2-2.tex),只用到了圆和直线:

```
\setlength{\unitlength}{1mm}
\begin{center}
\begin{picture}(35,32)
\put(20,7){\circle*{2}}
\put(20,7){\circle{14}}
\put(0.00,11){\line(6,1){35}}
\put(15.5,13.6){\line(3,5){4.5}}
\put(20,21){\line(1,-2){3}}
\put(20,21){\line(0,1){7}}
\put(13,28){\line(6,-1){14}}
\put(20,30){\circle{4}}
\end{picture}
\end{center}
```



所谓圆角矩形就是一个矩形的四角被四分之一圆代替. LaTeX对圆的直径有一定限制,圆角矩形四角上圆的直径在不超过矩形短边长度的条件下,总是尽量的大.这种图形也称为卵形线.除了完整的圆角矩形,二分之一或四分之一的圆角矩形也经常用到,对应的命令是

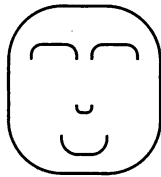
`\oval(宽度,高度)[部分]`

其中的 (宽度, 高度) 是圆角矩形的外切矩形宽度和高度. 定位命令 `\put` 或 `\multiput` 中的坐标是 整个圆角矩形对称中心的坐标. 可选项参数 部分 可以取如下值:

- `t top`, 上半部圆角矩形.
- `b bottom`, 下半部圆角矩形.
- `l left`, 左半部圆角矩形.
- `r right`, 右半部圆角矩形.
- `tl` 左上角四分之一圆角矩形.
- `tr` 右上角四分之一圆角矩形.
- `bl` 左下角四分之一圆角矩形.
- `br` 右下角四分之一圆角矩形.

对于圆来说, 没有画部分圆周的参数. 通过把圆角矩形的高和宽取成相同的值, 可以得到半个或四分之一圆周. 与圆的限制一样, 不能得到任意直径的部分圆周.

下面是用圆角矩形曲线画的一个图(见 6-2-3.tex).



```

\begin{picture}(20,22)
\thicklines
\put(10,11){\oval(20,22)}
\multiput(6,15)(8,0){2}{\oval(6,4)[t]}
\put(10,9){\oval(2,2)[b]}
\put(10,5){\oval(6,5)[b]}
\end{picture>

```

6.2.3 图形中的盒子

在绘图环境中, 将无框盒子和有框盒子的命令作了推广, 并增加了虚线框盒子, 它们是:

```

\makebox(宽,高)[位置]{文本}
\framebox(宽,高)[位置]{文本}
\dashbox{虚线长度}(宽,高)[位置]{文本}

```

其中(宽, 高)指定矩形盒子的宽度和高度, 均以 `\unitlength` 的值为单位. 可选项参数 位置 指定文本在盒子中的位置, 可取如下值:

- `t top`, 文本水平方向居中, 竖直方向靠在盒子顶部.
- `b bottom`, 文本水平方向居中, 竖直方向靠在盒子底部.
- `l left`, 文本竖直方向居中, 水平方向靠在盒子左边.
- `r right`, 文本竖直方向居中, 水平方向靠在盒子右边.
- `s stretch`, 文本竖直方向居中, 水平方向伸展充满盒子.

tl top left, 文本位于盒子的左上角.

tr top right, 文本位于盒子的右上角.

bl bottom left, 文本位于盒子的左下角.

br bottom right, 文本位于盒子的右下角.

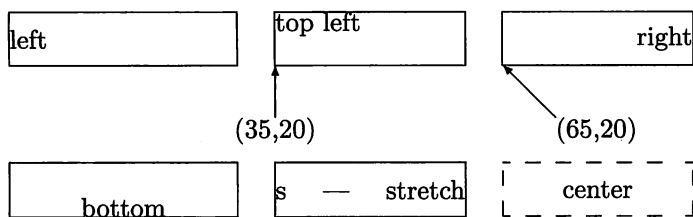
如果省略了可选项参数位置, 则文本在水平方向和竖直方向上都是居中地放置在盒子中. 可选项中有两个字母时, 顺序无关紧要, 例如 [tl] 与 [lt] 的作用相同.

命令中的虚线长度是组成虚线的小短线的长度, 两个小段线之间的空白也是这个长度, 以 `\unitlength` 的值为单位. 当盒子的宽度和长度是短线长度的整数倍时, 虚线框要好看一些.

将盒子命令作为图形元素放在 `\put` 或 `\multiput` 命令中时, 定位坐标指定了盒子基准点的位置, 也就是盒子左下角的位置. 下面是一个例子 (见 6-2-4.tex), 输入

```
\setlength{\unitlength}{1mm}
\begin{picture}(90,27)
\put(0,0){\framebox(30,7)[b]{bottom}}
\put(35,0){\framebox(25,7)[s]{s --- stretch}}
\put(65,0){\dashbox{2}(25,7){center}}
\put(0,20){\framebox(30,7)[l]{left}}
\put(35,20){\framebox(25,7)[tl]{top left}}
\put(65,20){\framebox(25,7)[r]{right}}
\put(35,13){\makebox(0,0)[t]{(35,20)}}
\put(35,13){\vector(0,1){7}}
\put(72,13){\makebox(0,0)[tl]{(65,20)}}
\put(72,13){\vector(-1,1){7}}
\end{picture}
```

输出为



图中的箭头及坐标指出了两个盒子的基准点坐标.

图形元素 `\makebox` 是个没框的盒子, 使用最多的情况是把它的宽度和高度都取为零, 使盒子变为一个点. 利用位置参数很容易确定文本与该点的相对关系, 从而容易把文本放在任何所希望的位置上. 图中两个无框盒子都是一个点, 矢量线的起点与盒子的基准点坐标相同, 由此可直观看出文本与基准点的位置关系.

为了使图形与上下相邻文本有较大的竖直间隔, 应在绘图环境的前后插入竖直间距命令, 例如插入 `\medskip`.

仔细观察图中的数据, 可以算出所有图形元素占据的范围就是绘图环境给出的尺寸 (90, 27). 如果把这个尺寸改为 (90, 37), 并把图形区域左下角的坐标由默认值 (0, 0) 改为 (0, -5), 就会使图形区域在高度方向增加 10mm 的空白, 而且空白被分在了图形的上方和下方, 个中原因不再解释了, 请读者自行分析.

6.2.4 图形中的文本

在图形中插入文本, 最简单的方法是在定位命令后面直接写上文本:

```
\put(x, y){文本}
```

这样的文本不能太长, 因为它不会被换行. 坐标 (x, y) 是文本行基准点的位置, 粗略地说就是文本所占区域左下角的位置.

为了容易把文本放置在指定位置, 更常用的方法是使用上一节介绍的长宽为零的无框盒子:

```
\put(x, y){\makebox(0,0)[位置]{文本}}
```

在文字模式或绘图环境中都可排版竖直堆叠文本, 命令为

```
\shortstack[位置]{一系列文本}
```

其中位置可取值是 l, r 或 c, 默认是 c, 表示一系列文本在其所占区域中的对齐方式. 所占区域的左下角是基准点, 定位命令 `\put` 或 `\multiput` 指定的坐标就是该基准点的坐标. 竖直堆叠文本行之间的间隔很小, 显得紧凑. 下面是一个例子 (见 6-2-5.tex), 输入

```
\begin{center}
\begin{picture}(63,15)
\put(0,0){\shortstack[l]{这是\\左对齐\\堆叠文本}}
\put(20,0){\shortstack[r]{这是\\右对齐\\堆叠文本}}}
\put(40,0){\shortstack{这是\\居中对齐\\堆叠文本}}
\put(60,0){\shortstack{A\\B\\C\\D}}
\end{picture}
\end{center}
```

输出为

这是	这是	这是	A
左对齐	右对齐	居中对齐	B
堆叠文本	堆叠文本	堆叠文本	C
			D

6.2.5 曲线

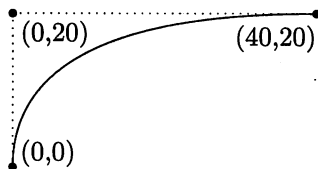
在 \LaTeX 中绘制曲线的命令是

```
\bezier{数}(x_1, y_1)(x_2, y_2)(x_3, y_3)
\qbezier[数](x_1, y_1)(x_2, y_2)(x_3, y_3)
```

它们都是画一条从 (x_1, y_1) 到 (x_3, y_3) 的 Bézier 曲线, 是用 数+1 个点画出来的。 (x_2, y_2) 是控制曲线弯曲程度的控制点: 从该点到两个端点的连线是曲线的两条切线。上面两条命令的区别是第二个命令中的画图点数为可选项, 如果省略该项, \LaTeX 会自动计算出合适的点数以画成一条光滑的 Bézier 曲线。保留第一条命令仅仅是为了和以前的 \LaTeX 2.09 版本兼容。下面的例子直观显示了 3 个点的关系 (见 6-2-6.tex), 输入

```
\begin{center}\begin{picture}(41,21)(-0.5,-0.5)
\put(0,0){\circle*{1}}\quad\put(0,20){\circle*{1}}
\put(40,20){\circle*{1}}\qbezier[20](0,0)(0,10)(0,20)
\qbezier[40](0,20)(20,20)(40,20)
\qbezier(0,0)(0,20)(40,20)
\put(1,0){\makebox(0,0)[b1]{(0,0)}}
\put(1,19){\makebox(0,0)[t1]{(0,20)}}
\put(40,18.5){\makebox(0,0)[tr]{(40,20)}}
\end{picture}\end{center}
```

输出为



图中的虚线是用 Bézier 曲线画成的, 这只要使命令中的 3 个点位于一条直线上, 并且指定较少的点数即可。利用这种方法也可以画出具有任意斜率的直线, 但要指定足够多的点数 (或不指定点数)。

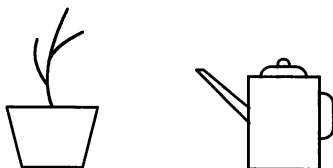
§ 6.3 子图

把图形中的部分图形当成一个整体处理, 这部分图形就称之为子图。子图可以由图形环境产生, 称为子图环境。这实际构成绘图环境的嵌套, 即在定位命令 \put 或 \multiput 中的图形元素是另一个绘图环境, 以生成子图。子图环境有自己的坐标系, 其定位命令中的坐标都是相对于该坐标系的。在子图环境中可以设置自己的

单位长度以及线的粗细,若不设定,则使用外部绘图环境的值. 当一个图形中含有几个图形时,使用子图环境可以减少定位差错,并且可以容易地调整改变相对位置. 下面是一个含有两个子图环境的例子,源文件缩行输入,便于看出每个子图环境的语句,容易检查错误. 输入是(见 6-3-1.tex)

```
\begin{center}
\begin{picture}(44,21)
\thicklines
\put(0,0){\begin{picture}(12,21)
\put(0,8){\line(1,0){12}}
\put(2,0){\line(1,0){8}}
\put(2,0){\line(-1,4){2}}
\put(10,0){\line(1,4){2}}
\qBezier(6,8)(4,13)(8,21)
\qBezier(5.3,11)(3,14)(4,17)
\qBezier(5.5,14)(6,16)(10,18)
\end{picture}}
\put(25,0){\begin{picture}(17,14.5)
\put(7,0){\framebox(9,12){}}
\put(7,8){\line(-6,5){6}}
\put(7,6){\line(-1,1){7}}
\put(0,13){\line(1,0){1}}
\put(16,7){\oval(4,6)[r]}
\put(11.5,12){\oval(6,3)[t]}
\put(11.5,13.5){\oval(1.5,2)[t]}
\end{picture}}
\end{picture}
\end{center}
```

输出为



子图可以保存在一个盒子中,以后可以整体调用这个盒子,不必再一一输入各条命令. 这个盒子称为子图盒子. 要保存和重复使用子图,首先应创建子图盒子名称:

```
\newsavebox{\子图盒子}
```

然后用下述命令将部分图形保存在名为子图盒子的子图盒子中:

```
\savebox{\子图盒子}(宽度,高度)[位置]{子图图形}
```

其中参数(宽度,高度)[位置]的含义与第81页\makebox中的含义一样. 参数子图图形可以是文本,也可以是一个子图环境,或者是一些绘图命令.

以后可以把命令

```
\usebox{\子图盒子}
```

当作图形元素放置在其他图形中的任何地方.

如果\savebox定义在一个环境中,那么随着环境的结束,它的值也就消失了. 如果定义在导言区,那么整篇文档都可使用这个子图盒子. 但要注意,子图盒子要占用分配给T_EX的内存空间,所以当不再使用子图盒子时,可将其内容重设为空,简单的办法是使用命令:

```
\sbox{\子图盒子}{}
```

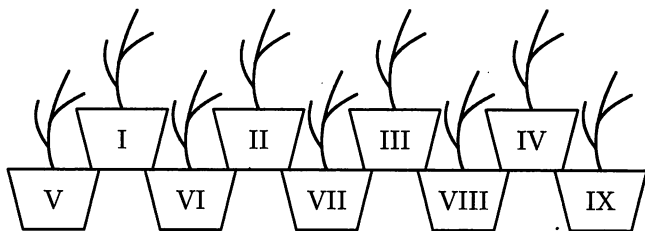
下面是使用子图盒子的例子,首先定义子图盒子名称,然后将子图保存在这个盒子中,这是由下述语句完成的:

```
\newsavebox{\flower}
\savebox{\flower}(12,21){%
\begin{picture}(12,21) \thicklines
\put(0,8){\line(1,0){12}} \put(2,0){\line(1,0){8}}
\put(2,0){\line(-1,4){2}} \put(10,0){\line(1,4){2}}
\qbezier(6,8)(4,13)(8,21) \qbezier(5.3,11)(3,14)(4,17)
\qbezier(5.5,14)(6,16)(10,18) \end{picture}}
```

下面是在一个图形环境中使用子图盒子(见6-3-2.tex).

```
\newcommand{\wrt}[1]{\makebox(0,0)[c]{#1}}
\newcounter{pot}
\setcounter{pot}{0}
\begin{center}
\begin{picture}(84,29)
\multiput(0,0)(18,0){5}{\usebox{\flower}}
\multiput(9,8)(18,0){4}{\usebox{\flower}}
\multiput(15,12)(18,0){4}{\addtocounter{pot}{1}\wrt{\Roman{pot}}}
\multiput(6,4)(18,0){5}{\addtocounter{pot}{1}\wrt{\Roman{pot}}}
\end{picture}
\end{center}
\sbox{\flower}{}
```

为了在图形中显示顺序数字, 设置了一个计数器 `pot`, 每用一次子图就使计数器加 1, 并且用大写罗马数字显示计数器的值. 显示结果为



因为以后不再使用子图盒子 `\flower`, 所以最后用一条命令释放子图盒子占用的内存.

子图不一定是用子图环境生成的, 也可以是单独的绘图命令. 下面例子定义了几个子图盒子, 存放的子图分别是流程图中的处理框、条件判断框和输入输出框, 这 3 个子图的坐标原点都放在了子图的中心. 另外定义了几个命令简化水平和垂直方向的直线和矢量线的输入. 源文件是 (见 6-3-3.tex):

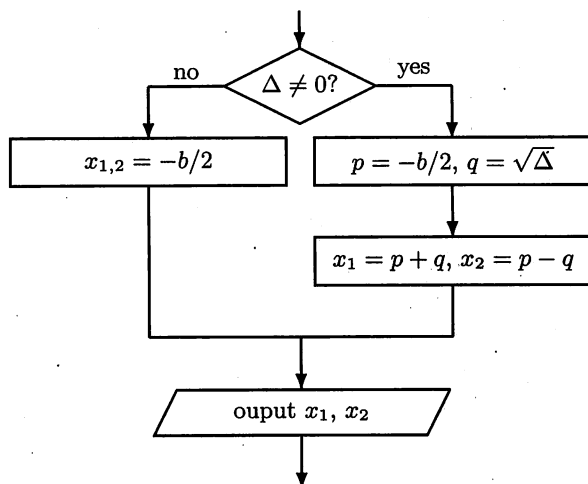
```
\newcommand{\wrt}[1]{\makebox(0,0)[c]{#1}}
\newcommand{\lline}[1]{\line(-1,0){#1}}
\newcommand{\rline}[1]{\line(1,0){#1}}
\newcommand{\uline}[1]{\line(0,1){#1}}
\newcommand{\dline}[1]{\line(0,-1){#1}}
\newcommand{\lvec}[1]{\vector(-1,0){#1}}
\newcommand{\rvec}[1]{\vector(1,0){#1}}
\newcommand{\uvec}[1]{\vector(0,1){#1}}
\newcommand{\dvec}[1]{\vector(0,-1){#1}}
\newsavebox{\condition}
\newsavebox{\process}
\newsavebox{\inputoutput}
\savebox{\process}(0,0){\thicklines
  \put(-18,-3){\framebox(36,6){}}
}
\savebox{\condition}(0,0){\thicklines
  \put(-10,0){\line(2,1){10}}
  \put(-10,0){\line(2,-1){10}}
  \put(10,0){\line(-2,1){10}}
  \put(10,0){\line(-2,-1){10}}
  \put(-10,0){\lline{10}}
  \put(-15,1){\makebox(0,0)[b]{no}}
  \put(10,0){\rline{10}}
```

```

\put(15,1){\makebox(0,0)[b]{yes}}
}
\savebox{\inputoutput}(0,0){\thicklines
\put(-19.5,-3){\rline{36}}
\put(-19.5,-3){\line(1,2){3}}
\put(19.5,3){\lline{36}}
\put(19.5,3){\line(-1,-2){3}}
}
\begin{center}
\begin{picture}(80,63)(0,-63)\thicklines
\put(40,0){\dvec{5}}
\put(40,-10){\usebox{\condition}}
\put(40,-10){\wrt{\Delta\ne0$?}}
\put(20,-10){\dvec{7}}
\put(20,-20){\usebox{\process}}
\put(20,-20){\wrt{$x_{1,2}=-b/2$}}
\put(60,-10){\dvec{7}}
\put(60,-20){\usebox{\process}}
\put(60,-20){\wrt{$p=-b/2,\,q=\sqrt{\Delta}$}}
\put(60,-23){\dvec{7}}
\put(60,-33){\usebox{\process}}
\put(60,-33){\wrt{$x_1=p+q,\,x_2=p-q$}}
\put(20,-23){\dline{20}}
\put(60,-36){\dline{7}}
\put(20,-43){\rline{40}}
\put(40,-43){\dvec{7}}
\put(40,-53){\usebox{\inputoutput}}
\put(40,-53){\wrt{ouput $x_1,\,x_2$}}
\put(40,-56){\dvec{7}}
\end{picture}
\end{center}

```

输出为



§ 6.4 天元的绘图功能

Tydraw 是肖刚设计的在 T_EX 环境里使用的绘图软件, 其特点是能利用 T_EX 画出精确的插图. 经陈志杰修改, 又增加了一些新的功能.

Tydraw 可以单独使用. 含有 Tydraw 命令的源文件不应该使用 .tyd 作为后缀, 因为这是 Tydraw 输出文件默认的扩展名. 源文件完成后, 执行下面所述的命令 (文件名是指带后缀的全名):

tydraw 输入文件名 [输出文件名]

注意按随书光盘的自动安装程序, tydraw.exe 被存放在 localtexmf\tytex 里, 因此如果经常使用 Tydraw 的话, 最好把上述路径加到机器的搜索路径里. 否则执行上述命令时会发生找不到执行文件的错误. 方括号表示 输出文件名 可以省略, 这时会自动输出扩展名为 tyd 的同名文件. 这个程序把 Tydraw 的命令转换成 T_EX 的命令, 其余内容丝毫不变. 然后再放到 T_EX 文件里或修改扩展名后用 T_EX 编译, 就能得到所要的图形. 如果 Tydraw 的命令有误, Tydraw 会显示一个出错信息, 在文件 tydraw.log 中会纪录一些简单的信息供查错之用.

如果在源程序中使用了 Tydraw, 则必须在前面的 (例如在导言区) 加入语句 \input tdw. Tydraw 最好嵌入在 L^AT_EX 的 picture 环境中使用, 因为 Tydraw 使用的长度单位是 1mm, 当 \unitlength 被设定为 1mm 后, Tydraw 与 picture 环境配合得天衣无缝. 因此最好在文件的导言区加入命令:

\setlength{\unitlength}{1mm}

在 picture 环境中, Tydraw 命令块的整体结构如下:

\put(a,b){\draw c,d,{...}}

这里 a, b, c, d 应该是 4 个数, Tydraw 所绘图形的参考点被安放在 picture 环境的坐标为 (a, b) 的点上, 而 $\{\dots\}$ 中的各条指令的原点则定位在相对于 Tydraw 所绘图形的参考点(左下角)的坐标为 (c, d) 的点上. 因此 Tydraw 图形原点在 picture 环境里的坐标是 $(a + c, b + d)$, 特别当 a, b, c, d 都取 0 时, Tydraw 的坐标系与 picture 环境里的坐标系一致.

$\{\dots\}$ 中的 Tydraw 命令如以下所列(注意所有的命令均不含 '\', 相互之间用逗号 ',' 分隔). 命令中的参数都可以使用表达式. 所谓表达式是指可使用符号: +, -, *, / 表示加减乘除(当无歧义时, 乘号 * 可省略, 例如 $100\sin(t)$); ^ 表示乘幂(注意底数不能等于 0, 当指数不是整数时, 底数必须是正数). 指数运算的级别最高, 然后是乘法和除法, 最后才是加减法, 与通常代数运算的惯例相同. $|\dots|$ 表示绝对值, e 和 pi 代表常数 e 和圆周率. 用 sin, cos, tg, arcsin, arccos, arctg, exp, ln, lg, sqrt, sh, ch, th 表示各种初等函数. 如果对运算的先后次序没有把握时, 可使用圆括号“(”与“)”, 但不能使用方括号或花括号.

Tydraw 命令:

1. 设定线条宽度: width(w) $w=0.1, 0.2, \dots, 1.0$. 一般可使用 0.2. 此命令可多次使用. 默认值为 0.1.
2. 画直线: line(x_1, y_1, x_2, y_2) 从 (x_1, y_1) 到 (x_2, y_2) 的直线.
3. 画平移直线: lines($x_1, y_1, x_2, y_2, dx, dy, n$) 从直线 $(x_1, y_1)-(x_2, y_2)$ 开始, 以 (dx, dy) 做平移向量, 画 n 条直线. 注意这里的 n 应是一个正整数.
4. 画折线: polygon($x_1, y_1, x_2, y_2, \dots, x_n, y_n$).
5. 画箭头: arrowhead(x, y, L, d_1, d_2) 或 arrowheadd(x, y, dx, dy, L, d_2) 以 (x, y) 作为箭尖, 画一个指向 d_1 度(或方向向量为 (dx, dy)) 的箭头, 箭头的边长为 L , 箭头的边与中心线的夹角为 d_2 度. 推荐值: $L=2, d_2=10$.
6. 画圆: circle(x, y, r) 圆心在 (x, y) , 半径为 r .
7. 点: point(x, y, d) d 为直径, $d \leq 18.8$.
8. 画长方形: box(x_1, y_1, x_2, y_2) 以 $(x_1, y_1), (x_2, y_2)$ 作为两个顶点的空心矩形.
9. 实心矩形: fill(x_1, y_1, x_2, y_2).
10. 参数方程定义的曲线: trace($t_1, t_2, f_1(t), f_2(t)$) 此命令是 Tydraw 的核心, 因为极大部分绘图软件都无法画出这类曲线, 这里曲线的参数方程为: $x=f_1(t), y=f_2(t)$, t_1, t_2 界定参数 t 的取值范围.
11. 虚线: dotted(n, p) 此命令以后的图形都被画成虚线. 如要再恢复实线模式, 可使用命令 dotted(0, 0). 其中 $n \leq 16$, 表示虚线的周期, 其长度单位是 1pt 左右. p 的二进制表示式就是每个周期内虚线的模式, 数字 1 表示黑点, 数字 0 表示空白. 当 $(n, p)=(6, 7)$ 时, 因为 7 的二进制表达式是 111, 因此这条虚线是 3pt 长的短线, 互相间隔 $6 - 3 = 3$ pt. 下面列出一些不同参数虚线的样本.

```

dotted(2,1) -----
dotted(4,3) -----
dotted(5,7) -----
dotted(6,7) -----
dotted(15,16381) -----

```

12. spline 曲线: `acurve(x1,y1,d1,...)` 以及 `curve(x1,y1,d1,...)` 规定在 (x_i, y_i) 处的切线角度为 d_i 度. 这两条命令画出的曲线略有不同, 用户可比较采用合适的一种.
13. 消隐: `hide(b1,L1,b2,L2,...)` 这条命令仅对紧跟在后面的命令有效. 表示在参数取值为 $b1, b2, \dots$ 的地方消隐长度为 $L1, L2, \dots$ 的弧. 最多消隐 32 处. 这里直线或 spline 线段的参数取值从 0 至 1, 圆周则从 0 至 2π .
14. 格点: `lattice(x0,y0,dx1,dy1,dx2,dy2,n1,n2,d)` 画 $n1 \times n2$ 个点, 以 $(x0,y0)$ 作为起点, 以 $(dx1,dy1)$ 和 $(dx2,dy2)$ 作为生成向量, d 是点的直径.
15. 网格: `net(t1,t2,u1,u2,f1(t,u),f2(t,u))` 画出具有两个参数的曲面网格: $x=f1(t,u)$, $y=f2(t,u)$. 网格的步长 (即参数 t, u 的增量) 都是 1.
16. 画阴影线: `shadow(t1,t2,dt,f1(t),g1(t),f2(t),g2(t))` 画出从点 $(f1(t), g1(t))$ 到点 $(f2(t), g2(t))$ 的直线, 其中参数 t 的取值从 $t1$ 至 $t2$, 以 dt 作为增量.
17. 缩放比: `ratio(r)` 默认的比值 $r=1$. 此命令设定 r 的值. 在此以后的命令中的参数均被放大 r 倍. 不过所有的角度都不受影响, `width`, `dotted`, `hide` 命令也不受影响. 此外 `lines` 中的 n , `arrowhead` 与 `arrowheadd` 中的 L , `point` 中的 d , `lattice` 中的 $n1, n2, d$, 以及 `trace`, `net`, `shadow` 中的参数变化范围 $t1, t2$ 均不受影响, 但参数方程的值都被放大 r 倍, 因此整个图形被放大了 r 倍, 如果你同时重新规定 `\unitlength` 的值为 r mm, 就能做到与 L^AT_EX 的图形同时放大.
18. 标注文字: `write(x,y)[位置]{标注内容}` 这条命令利用 `\makebox(0,0)[...]{...}` 把文字标注在图形上, 因此位置可选取 c, l, r, t, b 及其组合. 当 `ratio` 改变时, 这些文字的位置也会同步改变.
19. 三维曲线: `threed(θ, ϕ)` (θ, ϕ) 就是 Maple 立体图的视角: θ 是视线在 xOy 平面上的射影与 x 轴的夹角, ϕ 是视线与 z 轴的夹角. 例如 Maple 默认的视角是 $(45, 45)$, 正等测投影的视角是 $(45, 54.7)$, 正二等测投影的视角是 $(20.7, 70.5)$ 等. 从这条命令开始, 下列命令:

```

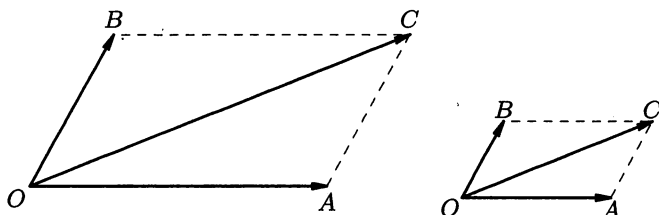
line(x1,y1,z1,x2,y2,z2)
lines(x1,y1,z1,x2,y2,z2,dx,dy,dz,n)
polygon(x1,y1,z1,x2,y2,z2,...,xn,yn,zn)
arrowhead(x,y,z,L,d1,d2)
arrowheadd(x,y,z,dx,dy,dz,L,d2)
point(x,y,z,r)
trace(t1,t2,f1(t),f2(t),f3(t))
write(x,y,z)[...]{...}

```

都变成了三维的平面图. 其他命令则不受影响. 如果要恢复平面图形的模式, 可使用命令 `twod`. 读者将会发现要改变视角十分容易, 唯一受影响的是曲线被遮盖的部分, 需要重新调整, 此外, 立体图形的包络也可能变化.

与 $\text{T}_{\text{E}}\text{X}$ 一样, `%` 被用来作为注解号, 从这个字符开始直至行尾都被忽略. 不过注解符 `%` 必须出现在完整的 `Tydraw` 命令的前面或后面, 如果出现在一条命令的中间, 就会出错.

我们先举一个向量相加的平行四边形法则的例子 (参见 6-4-1.ty).

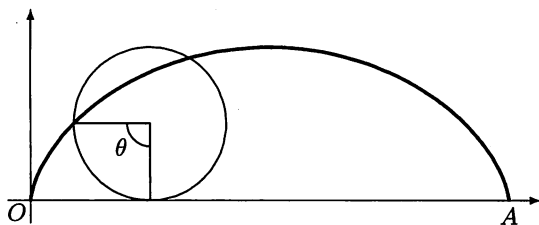


上图的源程序如下:

```
\begin{center}
\setlength{\unitlength}{1mm}
\begin{picture}(56,26)(-3,-3)\small
\put(0,0){\draw0,0,{width(.3),polygon(39,0,0,0,11,20),
line(0,0,50,20),arrowheadd(39,0,1,0,2,10),
arrowheadd(50,20,50,20,2,10),
arrowheadd(11,20,11,20,2,10),
dotted(6,7),width(0.1),polygon(39,0,50,20,11,20),
write(-0.5,-0.5)[rt]{$O$},write(39,-1)[t]{$A$},
write(11,21)[b]{$B$},write(50,21)[b]{$C$}}}
\end{picture}
\setlength{\unitlength}{0.5mm}
\begin{picture}(56,26)(-3,-3)\small
\put(0,0){\draw0,0,{width(.3),ratio(0.5),
.....
\end{picture}
\end{center}
```

由于使用了 `Tydraw` 命令 `ratio(0.5)` 以及把 `\unitlength` 设为 `0.5mm`, 不需改动源程序的其余部分, 就能使右边的图成为左边的一半大小. 而命令 `\small` 的作用是使得标注的字母变得小一些.

陈志杰主编的《高等代数与解析几何》中的插图都是用 `Tydraw` 画的. 下面是摆线图形及其 `Tydraw` 源程序 (参见 6-4-2.ty).



```
\begin{picture}(70,28)(-3,-3)\small
\put(-3,0){\vector(1,0){70}}
\put(0,-3){\vector(0,1){28}}
\put(0,0){\draw0,0,{width(0.4),
trace(0.01,2pi-0.01,10(t-sin(t)),10(1-cos(t))),
width(0.2),
circle(15.7,10,10),polygon(5.7,10,15.7,10,15.7,0),
trace(pi,1.5pi,15.7+3cos(t),10+3sin(t)),
write(-0.5,-0.5)[rt]{$O$},write(20pi,-1)[t]{$A$},
write(12.7,8)[rt]{$\theta$}}}
\end{picture}
```

摆线的一个拱的参数范围应该是0到 2π ,但是0与 2π 是摆线的奇点, Tydraw有可能出错,因此 trace 命令的参数范围改成0.01到 $2\pi - 0.01$,以避开奇点.

下面是绘制球面的立体图的例子(参见6-4-3.ty),左右两个图采用了不同的视角,左边是正等测投影的视角($45^\circ, 54.7^\circ$),右边是 Maple 默认的视角($45^\circ, 45^\circ$),除了 \threed 的参数不同外,其余命令完全相同,可见改变立体图的视角十分容易.当然读者会发现右边图形的虚实线画得不对,这可通过试验来调整.

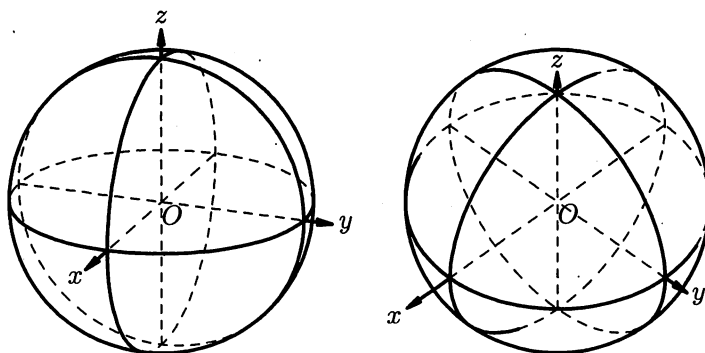
```
\begin{center}
\begin{picture}(44,44)(-20,-20)
\put(0,0){\draw0,0,{width(.4),
circle(0,0,20),
threed(20.7,70.3),
trace(-69.3*pi/180,110.7*pi/180,20cos(t),20sin(t),0),
trace(-0.366,pi-0.366,20sin(t),0,20cos(t)),
trace(-0.792,pi-0.792,0,20sin(t),20cos(t)),
line(20,0,0,28,0,0),
line(0,20,0,0,24,0),
line(0,0,20,0,0,24),
arrowheadadd(28,0,0,1,0,0,2,10),
arrowheadadd(0,24,0,0,1,0,2,10),
arrowheadadd(0,0,24,0,0,1,2,10),
write(29,0,0)[rt]{$x$},
```

```

write(0,25,0)[l]{$y$},
write(0,0,25)[b]{$z$},
write(0,0,-0.5)[lt]{$O$},
dotted(6,7),width(.2),
trace(110.7*pi/180,290.7*pi/180,20cos(t),20sin(t),0),
trace(2pi-0.366,pi-0.366,20sin(t),0,20cos(t)),
trace(2pi-0.792,pi-0.792,0,20sin(t),20cos(t)),
line(-20,0,0,20,0,0),
line(0,-20,0,0,20,0),
line(0,0,-20,0,0,20)}}
\end{picture}
\quad
\begin{picture}(44,44)(-20,-20)
\put(0,0){\draw0,0,{width(.4),
circle(0,0,20),
threed(45,45),
.....
\end{picture}
\end{center}

```

输出为



§ 6.5 浮动表格和图形

`picture` 环境和 `tabular` 环境生成的图形或表格通常是插入在当前位置, 也就是输入的位置. 如果当前页的剩余空间不够, 图表将被移动到下一页, 当前页就会出现很大的空白, 于是需要人工调整图表与前后文本的位置. 一处的调整可能影响后面多处的调整, 文稿有所修改时又要重新调整, 非常不便. \LaTeX 提供了浮动图表的功能, 遇到图表在当前位置放不下时, 会在一定的原则下自动调整图表的位置, 大大减轻了人们的工作量.

浮动图形和浮动表格环境是

```

\begin{figure}[位置] 图形 \end{figure}
\begin{figure*}[位置] 图形 \end{figure*}
\begin{table}[位置] 表格 \end{table}
\begin{table*}[位置] 表格 \end{table*}

```

对于单列页面格式, 带 * 号的环境与标准环境(无 * 号的环境)作用相同. 带 * 号的环境只适用于双列页面格式, 它使得图形或表格占据两列, 而不是正常情况的一列.

可选项 位置 由零个到 4 个字母组成, 这些字母是: h (here, 当前位置), t (top, 页芯顶部), b (bottom, 页芯底部), p (page, 单独一页). 上述字母或字母组合可与符号 ! 连用, 它会去掉对浮动所加的一些限制. 如果不写可选项, 默认是 t b p. 本书中带有图号的图形实际是如下输入的:

```

\begin{figure}[!htbp]
\centering
\begin{picture}
.....
\end{picture}
\caption{.....\label{.....}}
\end{figure}

```

位置选项字母的顺序是无关紧要的, L^AT_EX 总是按 h→t→b→p 的次序处理选项中的字母.

当有很多浮动体时, L^AT_EX 把表格与图形分别放在两个“先进先出”的线性队列中分别处理. 在每个队列中, 根据前后次序按着位置选项进行安排. 如果给定的位置选项不合适(例如只给出 h 选项), 在给定的位置安排不了, 它就会成为障碍, 堵塞整个队列, 这将把所有后续队列成员都推到了文档末尾.

命令

```

\clearpage      或      \cleardoublepage

```

立即安排浮动体队列中所有剩下的成员, 并开始新的一页, 其中第二个命令开始新的奇数页.

在浮动环境内使用命令

```

\caption[短标题]{标题\label{标签}}

```

可以给浮动体添加标题并自动编号. 如果别处不引用这个编号, 则不必写 \label 命令.

命令

```

\listoffigures      或      \listoftables

```

分别排印插图目录和表格目录, 如果给出了[短标题], 则目录中出现的就不再是标题而是短标题.

L^AT_EX 自动在标题编号前添加英文前缀 “Figure” 或 “Table”, 为了符合中文习惯, 可在正文中浮动环境和图表目录出现之前写命令

```
\renewcommand*\figurename{图}  
\renewcommand*\tablename{表}  
\renewcommand*\listfigurename{插图目录}  
\renewcommand*\listtablename{表格目录}
```

第七章 自定义与改错

本章首先将介绍如何定义你自己的命令,使得命令变得更容易记忆,或者使它们更合你的心意.同时你还可以根据自己的需要改变各种计数器以及长度的值,从而在设计版面方面拥有更多自主权. \TeX 的特点就是用户拥有极大的自由度,它实际上是一种编程语言,你了解得越深入,能做的事越多.当然作为基本篇,我们只介绍一些最表层的用法.在本章的后半部分简要地介绍了 \TeX 的出错信息以及警告信息,告诉你遇到这类信息时的处理原则.本章也是基本篇的结束,希望读者在学完本章后能够多多实践,然后当感到有提高的必要时,再去有的放矢地阅读提高篇,最终成为 \TeX 的能手.

§ 7.1 自定义命令

\LaTeX 本身提供的定义命令的语句有以下几个:

```
\newcommand{\新定义的命令名}[参量个数]{命令内容}  
\newcommand*{\新定义的命令名}[参量个数]{命令内容}  
\renewcommand{\重新定义的命令名}[参量个数]{命令内容}  
\renewcommand*{\重新定义的命令名}[参量个数]{命令内容}
```

这里参量个数不能超过 9 个.带星号“*”的命令为“短命令”——命令内容中不能出现段落,即不能有分段命令“ \par ”或空行.

很明显,前两个语句是定义原来没有的命令,后两个语句则是对原来已有定义的命令重新定义.这样做当然出于系统安全的目的.因为很多时候你并不知道所选定的命令名是否已经定义过,如果在无意之中把已经定义过的命令重新定义,就有可能产生无法预料的后果,而你甚至不知道错误起因于何处.现在当你新定义一个已经有定义的命令时, \LaTeX 在编译时会有出错信息,提醒你改用别的名字.

命令定义语句的原理很简单:每当 \LaTeX 在编译时遇到一条命令,就会寻找这条命令的定义语句,然后把这条命令用命令内容来替代,如果带有参数,则把参数的值代入,替换完成后,再对新的语句作编译.因此命令语句本质上就是替换,至于复杂的命令所引起的递归嵌套等问题,就超出本书的范围了.

自定义命令的第一个用处是可以节省输入的工作量,或帮助记忆命令.例如为打印希腊字母 ε ,需要输入命令“ \varepsilon ”,既长又容易出错.如果它在文章中经常被用到,那么可以给它取一个简称“ \eps ”或“ \e ”,即输入命令

```
\newcommand*{\e}{\varepsilon}
```

自此以后, L^AT_EX 一看到 “\e” 就会自动把它替换成 “\varepsilon”. 不过为了单独输出 ε , 还是必须输入 $\$ \backslash e \$$. 为了省掉这两个美元号, 我们可以把定义改为

```
\newcommand*{\e}{\varepsilon}
```

但又会产生新的问题: 当 ε 出现在数学模式, 例如 $|x| < \varepsilon$ 中时, 如果输入 $\$ |x| < \backslash e \$$, 经替换后变成 $\$ |x| < \backslash \varepsilon \$$, 使 $\backslash \varepsilon$ 出现在数学模式以外, L^AT_EX 就会产生出错信息. 为了使这个命令不论在数学模式以内还是以外都能使用, 可利用 L^AT_EX 的命令 $\backslash ensuremath$, 用法如下:

```
\newcommand*{\e}{\ensuremath{\varepsilon}}
```

又如 L^AT_EX 的命令都是来源于英语, 这对于精通英语的人来说当然能帮助记忆, 但对于英语不太好的人就会感到难记, 只要拼错一个字母, 就会出错通不过编译, 令人恼火. 现在就可以自己给它换一个易记的名字. 例如把 “\triangle” 改名为 “\SJX” (源自 “三角形” 的汉语拼音), 既短又好记. 为此只要输入:

```
\newcommand*{\SJX}{\ensuremath{\triangle}}
```

都采用大写字母是为了避免重名, 因为 L^AT_EX 的原始命令大都是小写字母.

复杂一点的命令还带有参量, 例如以下命令:

```
\newcommand*{\anvec}[2]{\ensuremath{\#1_1,\ldots,\#1_{\#2}}}
```

方括号中的数字 2 表明新命令 “\anvec” 带有两个参量, 它们在后面用花括号括起来的命令内容中分别表示为 “\#1” 与 “\#2”.

在具体使用命令 $\backslash anvec$ 时, 必须把这两个参量分别用花括号括起来, 紧跟在命令 $\backslash anvec$ 的后面. 所谓 “紧跟”, 是指它们之间除了空格以及至多一个换行外, 没有别的字母插入. 此外当参量只含一个字母的话, 也可把花括号省去. 例如下面 $\backslash anvec\{u\}\{n\}$ 与 $\backslash anvec\ u\ n$ 有相同效果, 经替换后都相当于 $\$ u_1,\ldots,u_n \$$, 最后产生输出 u_1,\ldots,u_n . 但是如果我们输入 $\backslash anvec\{u\}\{n+1\}$, 经替换后变成 $\$ u_1,\ldots,u_{n+1} \$$, 输出是 u_1,\ldots,u_n+1 , 而不是我们所期望的 u_1,\ldots,u_{n+1} . 问题显然出在第二个参量 $n+1$ 表面上看来是放在花括号内, 而实际上在替换时要去掉一层花括号, 就变成没有括号了. 为了避免这种差错, 我们可以在使用命令时多加一层括号, 输入为 $\backslash anvec\{u\}\{\{n+1\}\}$, 或者在定义时多加一重括号, 如下所示:

```
\newcommand*{\anvec}[2]{\ensuremath{\#1_1,\ldots,\#1_{\{\#2\}}}}
```

这样就万无一失了. 希望大家重视这个问题, 一个保险的办法就是给命令内容中出现的参量一律加上括号, 变成 $\{\#1\}, \ldots, \{\#9\}$, 罕见的例外是当参量用于条件语句的比较表达式时, 多加的一层花括号会导致非意料的结果.

最后说一下自定义命令的作用范围. 如果定义语句出现在导言部分, 那么这个命令对整个文件都有效, 你在这个定义语句出现以后的任何地方修改定义都必须使用 $\backslash renewcommand$. 而在文件的正文中, 各种环境以及由 $\{\}$ 构成的集团又使得文件内部被分成许多层次. 外层的命令在内层继续有效, 而内层定义的命令的作用范围是局部的, 一旦退出这个层次, 这个命令就不起作用了. 因此在内层修改外层的命

令时必须用 `\renewcommand`, 但一旦退出这个层次, 这个命令又自动恢复原先的意义. 一个在内层被定义过的命令如果在它的作用范围之外再次被定义, 则仍需使用命令 `\newcommand`, 这应该是很自然的. 因此在定义新的命令时要注意它的作用范围, 再决定用作局部定义还是整体定义. 一般我们可以把自己常用的定义统一放在文件的导言部分, 以利维护.

§ 7.2 给计数器和长度赋值

7.2.1 计数器

L^AT_EX 内部有不少计数器, 我们已经接触到的有以下一些:

part	subsection	subparagraph	footnote
chapter	subsubsection	page	
section	paragraph	equation	

注意计数器的名称没有倒斜线. 它们的意义从字面上就能看出, 例如 `page` 计数器就存放着当前页的页码. 如果我们需要改变计数器的值, 则可使用下列命令:

```
\setcounter{计数器名称}{数字}
\addtocounter{计数器名称}{数字}
\stepcounter{计数器名称}
```

这里的数字都应是整数, 其中 `\setcounter` 是给计数器赋值, `\addtocounter` 是把指定的值加到计数器上, `\stepcounter` 则把指定计数器的值加 1, 同时使下属计数器的值置零. 例如执行 `\stepcounter{section}` 后, `section` 计数器的值增加了 1, 同时它下面的计数器 `subsection`, `subsubsection`, `paragraph`, `subparagraph` 都被置零.

用户可以自己定义一些新的计数器, 命令为

```
\newcounter{新计数器名}[上级计数器名]
```

如果给出了 [上级计数器名], 则每当对上级计数器使用了命令 `\stepcounter`, 新计数器就被置零.

下述命令可以将一个计数器的值当作一个整数处理, 用在其他使用数值作为参数的命令中:

```
\value{计数器}
```

计数器的值可以用不同的数字样式显示, 命令为

```
\数字样式{计数器}
```

数字样式有如下几种:

<code>\arabic</code>	显示为阿拉伯数字(如 1, 2, 3, 4 等)
<code>\roman</code>	显示为小写罗马数字(如 i, ii, iii, iv 等)
<code>\Roman</code>	显示为大写罗马数字(如 I, II, III, IV 等)
<code>\alph</code>	显示为小写拉丁字母(如 a, b, c, d 等)
<code>\Alph</code>	显示为大写拉丁字母(如 A, B, C, D 等)
<code>\fnsymbol</code>	显示为脚标符号(*, †, ‡, §, ¶, , **, ††, ‡‡)

由于只有 26 个小写(或大写)拉丁字母, 因此在数字样式命令 `\alph` 和 `\Alph` 中, 计数器的值应位于 1 到 26 之间. 由于只有 9 个脚注符号, 所以使用数字样式命令 `\fnsymbol` 时, 计数器的值应位于 1 到 9 之间.

系统中定义了一些形如

`\the 计数器名称`

的命令, 用于按指定的数字样式显示计数器的当前值, 例如输入“本页是第`\, \thepage\,`页”, 结果为“本页是第 100 页”. 但有些这类命令的内部定义会将计数器的值与它的上级计数器的值组合起来, 例如若 `\thesection` 的定义为

```
\newcommand*{\thesection}{\thechapter.\arabic{section}}
```

那么输入“这是第`\, \thesection\,`节”, 则结果为“这是第 7.2 节”. 如果重新定义 `\thesection` 为

```
\renewcommand*{\thesection}{\S\thechapter-\roman{section}}
```

则结果为“这是第 §7-ii 节”.

7.2.2 长度

我们遇到过的长度命令有

<code>\arraycolsep</code>	<code>\parindent</code>	<code>\textheight</code>
<code>\baselineskip</code>	<code>\parskip</code>	<code>\textwidth</code>
<code>\bigskipamount</code>	<code>\smallskipamount</code>	
<code>\medskipamount</code>	<code>\tabcolsep</code>	

长度值应该是带有单位的长度, 例如 3mm 或者别的已有定义的长度的倍数, 例如 `0.5\textwidth`, `-\smallskipamount` 等.

用户可以自定义新长度命令:

`\newlength{\新长度命令}`

为了改变某些长度的值, 可以使用以下命令:


```
\setlength{\长度命令}{长度值}
\addtolength{\长度命令}{长度值}
```

第一条命令是给长度命令指定一个值, 第二条命令是给已有的长度增加长度, 若增加的是负值, 实际上就是减少长度. 灵活使用上述两个命令, 可以对一个长度作加减乘除运算, 例如下述命令的结果产生了一个长度 `\tmplength`, 它等于正文宽度减去段首缩进量后, 剩余宽度的 $1/4$:

```
\newlength{\tmplength}
\setlength{\tmplength}{\textwidth}
\addtolength{\tmplength}{-\parindent}
\setlength{\tmplength}{0.25\tmplength}
```

下述三个命令将长度命令的值分别设置为给定文本的宽度、高度和深度, 高度和深度的含义参见第 109 页.

```
\settowidth{\长度命令}{文本}
\settoheight{\长度命令}{文本}
\settodepth{\长度命令}{文本}
```

若想编译时在屏幕上显示某些长度的值, 可用命令:

```
\showthe\长度命令
```

例如想查看当前 `\tabcolsep` 的值, 可在原稿上写命令 `\showthe\tabcolsep`, 编译到此处屏幕上会显示这个命令的值及一个问号, 回车后就继续编译. 显示值不会插入到排版结果中.

§ 7.3 出错信息

L^AT_EX 的出错信息有两个来源, 一部分来自 L^AT_EX 本身, 还有一部分来自底层的 T_EX. 一个 L^AT_EX 错误可能会带来一系列 T_EX 的错误信息, 这是因为 L^AT_EX 的操作是位于 T_EX 之上的.

下面我们先看一个简单的例子 (7-3-1.tex):

```
\documentclass{article}
\begin{document}
The last words appear in \textbf{bold face}.
\end{document}
```

这里有一个错: `\textbf` 被误输入成 `\txetbf`. 用 LaTeX 编译时, 屏幕上会显示以下出错信息:

```
! Undefined control sequence.
1.4 The last words appear in \txetbf
                                   {bold face}.
?
```

这个出错信息是 T_EX 发出的, 因为 L^AT_EX 发出的出错信息的开头是

! LaTeX Error:

当然对大部分用户来说没有必要去区分它们. 这个出错信息告诉我们它遇到了“未被定义的控制序列”, 第 2 行的 1.4 指明错误出在源文件的第 4 行, 而且它刚刚把“\txetbf”读进去, 接在下一行的“{bold face}.”则是它将要读入的内容. 第 4 行的问号则是等待我们的处理意见.

如果我们再输入一个问号, 屏幕屏幕显示下面一段话:

```
Type <return> to proceed, S to scroll future error
messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore the next 1 to 9 tokens of input,
H for help, X to quit.
?
```

我们的答复有以下几种选择:

1. 按“回车”键, 让 T_EX 按预定的程序去处理各类错误, 同时继续往下编译;
2. 键入“S”再回车, 进入滚动模式 (scroll mode), T_EX 将不再停顿, 一直往下编译, 同时不断显示遇到的出错信息, 直到最终停止. 这相当于对所有的出错信息都按回车键;
3. 键入“R”再回车, 进入运行模式 (run mode), 这种模式与滚动模式类似, 但级别更高, 即在滚动模式会中断的地方 (例如在遇到输入文件的命令, 但没有提供文件名时) 它也不停止;
4. 键入“Q”再回车, 进入安静模式 (quiet mode), 这种模式同 R 模式, 只是不在屏幕上显示任何出错信息;
5. 键入“I”再用键盘输入你想插入的字符串, 例如遇到上述例子的错误, 可以输入“I\textbf”再回车, 就能继续编译, 并得到正确的结果. 不过请注意, 源文件并没有改动, 如果不修改源文件, 下次编译时仍会在这里出错. 如果你输入“I\stop”再回车, 则 T_EX 会停止编译, 并输出到当前页为止的 dvi 文件;
6. 键入一个小于 100 的正整数再回车, 则会删除接下去要输入的这么多个符号 (控制字或控制符算作 1 个符号), 并停下来等待你的进一步指示;

7. 键入“H”再回车, 这时屏幕会显示一些帮助信息, 提示你错误的可能原因及处理方法. 不过千万别寄托太大的希望, 经验证明, T_EX 的帮助信息往往解决不了什么问题;
8. 键入“X”再回车, T_EX 停止编译, 而且当前页不被输出到 dvi 文件;
9. 键入“E”再回车, 效果同“X”, T_EX 停止编译, 如果 T_EX 的编译程序与某个文字处理程序有关联的话, 就会自动进入文字处理程序, 并把光标移到出错的那一行.

注意上面输入的字母不分大小写, 都有相同效果.

L^AT_EX 的出错信息可以用同样的方式处理. 所有的出错信息都记录在与源文件同名, 但扩展名是 log 的文件上, 因此你不用为面对屏幕上飞滚而过的出错信息发愁, 只要打开 log 文件, 就可以一条条浏览出错信息并作适当处理. 我们的经验是遇到出错就按“Q”, 任其继续, 等停止后再根据 log 文件的纪录进行修改. 一般比较多的错误是输入打字错误, 还有常犯的错误是漏掉进入数学模式的美元号 \$ 或者花括号 {} 不配对. 有时 T_EX 提示的错误原因不一定是真正的原因, 而仅是诱发的错误, 因此初学者要仔细审读原文, 查出错误的真正原因, 等以后经验丰富了, 查错也变得容易了.

§ 7.4 警告信息

这里只介绍两类最重要的警告信息: Overfull 以及 Underfull.

如果在水平方向超长, T_EX 就会发出如下警告信息:

```
Overfull \hbox (10.00104pt too wide) in paragraph at lines 5--6
```

告诉你源文件第 5 至 6 行的地方超长了 10.00104pt. 如果是 draft 模式, 那么在 dvi 文件的屏幕显示或打印时会在超长的地方出现一个黑方块, 以引起注意. 对于西文的原稿, 可用插入隐含连字符“\-”的方法帮助 T_EX 换行, 以避免超长, 也可以人工干预, 用 \linebreak 强迫换行的方法解决问题. 如果超长不多, 也可不予处理, 在定稿时改用 final 模式, 使得黑方块消失, 一般不注意是看不出来的.

如果在竖直方向超长, T_EX 会发出如下警告信息:

```
Overfull \vbox ...
```

告诉你页面的下端超长了. 如果超得不多, 一般可不予处理, 如果超得多了, 就要采取调整版面内容或强迫换页的方式来解决.

如果看到以下警告:

```
Underfull \hbox (badness 5504) in paragraph at lines 5--7
```

就是提醒你这里发生了太空松的状况, 其严重程度是 5504. 严重程度以 10000 封顶, 如果达到 10000 就是非常严重了. 空松往往是由不适当的强迫换行引起的, 一般需要人工干预加以调整. 如果严重度不大, 也可以不处理.

如果在竖直方向出现空松, 会出现如下警告:

Underfull \vbox ...

读者同样可根据其严重程度决定是否需要处理.

提 高 篇

第八章 文字模式的高级技巧

§ 8.1 使文本居左或居右

类似于居中对齐环境(见第 32 页), 居左对齐环境

```
\begin{flushleft}
  第一行\\
  第二行\\
  .....
  第末行\\
\end{flushleft}
```

使得文本靠左对齐. 对于超长文本自动分行, 不进行断词并且使单词间距相同.

类似于 `\centering`, 在一个环境内部, 可以使用声明

```
\raggedright
```

使后继文本左对齐, 声明的作用到环境结束时为止. 同样可以使用 `\\` 强制分行. 这个声明的字面含义是“右边可以不齐”, 实际作用是“左边一定对齐”.

居右对齐环境和居右对齐声明分别是

```
\begin{flushright}
  第一行\\
  第二行\\
  .....
  第末行\\
\end{flushright}
```

```
\raggedleft
```

除了是使文本右对齐外, 其他与左对齐环境和声明是类似的.

紧跟在对齐环境后的文本如果与环境之间没有用空行分开, 则认为与环境中的文本属于同一个段落, 即此时环境后的第一行没有段首缩进, 但与环境中的文本仍有额外的竖直间隔.

§ 8.2 引文

在文章中的大段引文常常是另起一段, 并且将段落两边向内缩进一定距离. 下面两个环境都可实现这个要求:

```
\begin{quotation}  
  文本  
\end{quotation}
```

```
\begin{quote}  
  文本  
\end{quote}
```

其中文本可以很短, 只有几个字, 也可很长, 甚至包含几个段落.

上述两个环境有如下区别: 第一种环境中的段落像正常段落一样, 有首行缩进(前提是通常的段落是有首行缩进的), 段落之间具有正常的间距. 第二种环境并不自动首行缩进, 而是自动增大了段落间距.

对于上述两种环境, 在显示文本上方和下方都被自动插入了额外间隔.

排版西文诗歌、韵文时, 也是需要两边缩进, 所用环境是

```
\begin{verse}  
  诗歌、韵文  
\end{verse}
```

排版效果类似于 `quote` 环境, 即段落(诗歌的小节)首行不缩进, 段落间隔加大. 但在诗歌环境中对超长的行自动分行时, 分成的几行不是靠左对齐而是左端“悬挂”, 即后面几行比第一行要向右缩进一点距离.

上述引文及诗歌环境可以相互或自我嵌套, 最多嵌套6层. 每个内层环境都比外层环境更向内缩进.

§ 8.3 抄录

有时我们希望不对某些文稿进行格式处理, 而是按照输入的文本原样输出, 例如原样输出一段缩排的程序, 所有空格和空行也要保持原样, 这可使用下列抄录环境来实现:

```
\begin{verbatim}  
  文本  
\end{verbatim}
```



```
\begin{verbatim*}
  文本
\end{verbatim*}
```

其中文本中的西文会用打字机字体按原样输出, 文本内的控制命令也被当成普通字符串原样显示出来. 在CJK环境中, 抄录环境内的汉字也原样输出, 字体为进入抄录环境前的汉字字体.

上述两种抄录环境区别是显示的空格不同, 不带*号(称标准形式)的环境用空白显示空格, 带*号的环境用□显示空格.

对于不超过一行的文本, 两个抄录环境可以简化成两个抄录命令:

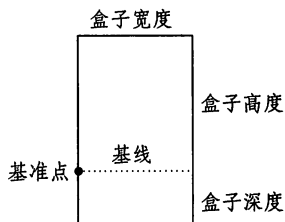
```
\verb|文本|
\verb*|文本|
```

带*号的命令与带*号的环境一样用可见的符号□显示空格. 表示定界符的竖线“|”可以换成除*号和空格以外的任何没有出现在文本内的符号, 左右定界符必须使用同一个符号, 例如\verb(文本(. 如果漏掉了右面的定界符, 或没有与左面相同的定界符, 则抄录的范围直到当前输入行的结束.

需要注意几点: 一是抄录环境和抄录命令都不能用作其他命令的参数; 二是不用抄录环境而用抄录命令时, 文本不能多于一行(在文稿中必须输入在同一行上, 排版输出也是在同一行上, 因此不要过长); 三是键盘上的重音号“`”和单引号“'”在抄录环境或抄录命令中仍是显示为英文单引号“'”和“'”.

§ 8.4 盒子

在T_EX中一切都是盒子, 单个字符是盒子, 若干字符盒子排成一行构成大一点的行盒子, 若干行盒子堆叠成段落盒子或页盒子. T_EX排版的过程实际就是在构造盒子和排列堆叠盒子. 盒子之间插入了可伸缩的弹性长度. 除了上述这些隐含的盒子, 还有一些显式生成盒子的命令, 后面会详细介绍. 每种盒子无论是带框的还是无框的, 都占据了一个矩形区域, 如下图所示:



图中的虚线称为盒子的“基线”, 当盒子相互左右边靠边排列时, 它们的基线总是位于同一条水平线上. 基线的左端点(图中的小黑点)称为盒子的“基准点”或“参考点”. 当盒子上下排列时, 它们的基准点总是位于同一条竖直线上.

每个盒子都有3个尺寸: 高度、深度和宽度, 注意盒子高度不是整个矩形的高度而是基线上方的高度. 矩形的高度称为总高度, 等于盒子高度与盒子深度之和. 这些尺寸用下列命令表示:

<code>\width</code>	表示盒子的宽度
<code>\height</code>	表示盒子的高度
<code>\depth</code>	表示盒子的深度
<code>\totalheight</code>	表示盒子的总高度

L^AT_EX 用户可以使用命令构造三种类型的盒子: LR 盒子、段落盒子和标尺盒子. LR (左-右) 盒子中包含的是从左到右的排版对象, 它们排在同一行上, 不能被分割成多行. 段落盒子由竖直堆叠的行组成. 标尺盒子 (或称划线盒子) 是一个实心矩形, 通常用于画水平或竖直线段.

8.4.1 LR 盒子

下列4个命令都可以创建LR盒子:

```
\mbox{单行文本}
\fbox{单行文本}
\makebox[宽度][位置]{单行文本}
\framebox[宽度][位置]{单行文本}
```

单行文本的含义是它不能被分行, 但它内部可以含有作为一个整体的含有多行文本的环境或盒子.

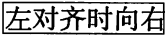
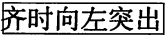
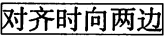
前两条命令生成的LR盒子, 其宽度等于单行文本排版后的宽度, 通常称为自然宽度. 这两个命令的区别是上一个命令产生的盒子没有边框, 下一个则产生带边框的盒子, 把单行文本包围起来.

后两条命令产生的LR盒子的宽度由可选项宽度给定. 另一个可选项位置指定单行文本在盒子中的排放位置, 它可取如下的值:


l	文本的左端紧靠盒子的左端
r	文本的右端紧靠盒子的右端
s	伸展单词间隔使文本撑满盒子
缺省	文本的中点对齐在盒子的中点

对于后两个命令, 如果两个可选项都缺省, 那就等同于前两个命令了. 如果省略一个, 则只能省略[位置], 此时表示盒内文本居中排列. 后两个命令的区别也是上一个产生无框盒子, 下一个产生有框盒子.

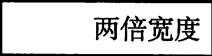
当指定的宽度小于单行文本的自然宽度时, 文本就会超出盒子, 根据位置的值很容易确定文本是在盒子的左边、右边还是两边突出出来. 例如

`\framebox[6em][l]{左对齐时向右突出}` 结果为  左对齐时向右突出
`\framebox[6em][r]{右对齐时向左突出}` 结果为  右对齐时向左突出
`\framebox[6em]{中间对齐时向两边突出}` 结果为  中间对齐时向两边突出

上述例子(参见 8-4-1.tex)仅仅为了形象地说明对齐与超出方向的关系,没有多少实用价值.但若改用 `\makebox` 并指定宽度为零,则在 `picture` 环境中就有用武之地了,用它很容易在指定的作图点生成居中对齐或左(右)对齐的文本.用它也可以使两部分文本重叠,例如

`\makebox[0pt][l]{/\backslash}` 结果为 

命令中的宽度可以利用盒子的自然尺寸来定义,所谓自然尺寸就是相当于用 `\mbox` 产生的盒子的尺寸,不是人为指定的尺寸.例如

`\framebox[2\width][r]{两倍宽度}` 结果为  两倍宽度

其中的 `\width` 代表的就是文本的自然宽度,前面的数字 2 是乘法因子.

表示盒子尺寸的命令可用在 LR 盒子的宽度选项中或用在子段盒子的高度选项中,一般不要用在其他地方,以免产生出错信息.

盒子框线与内部文本之间有一定间隔,命令 `\fboxsep` 保存这个间隔值.盒子的框线有一定的粗细,其值保存在命令 `\fboxrule` 中.可按需要修改这两个值:

```
\setlength{\fboxsep}{尺度}
\setlength{\fboxrule}{尺度}
```

上述设置尺度的命令放在导言区时,对整个文档起作用,若放在某个环境或分组内,则仅在局部范围起作用.在前面演示文本超出盒子的例子里,上述两个值分别设置为 1pt 和 0.2pt,其他例子使用的是默认值 (5pt 和 1pt).

如果某些文本在文档中多次出现,可以用一个盒子把它保存起来,以后使用就方便了.首先用命令

```
\newsavebox{\盒子名}
```

创建一个名为 `\盒子名` 的盒子.注意名字不能与任何已有的命令重名,并要符合命令的命名规则(斜线后面只能跟字母).然后就可使用命令

```
\sbox{\盒子名}{文本}
\savebox{\盒子名}[宽度][位置]{文本}
```

把文本的内容保存在盒子中.各个参数的意义与建立 LR 盒子的命令参数相同.以后随时可用命令

```
\usebox{\盒子名}
```

把保存在盒子中的文本当作一个整体插入到文档中. 这样的盒子是无框的, 若想加框, 可写成 `\fbox{\usebox{盒子名}}`.

利用 $\text{T}_{\text{E}}\text{X}$ 命令可以设置或改变盒子的宽度、高度和深度:

```
\wd\盒子名=长度   \ht\盒子名=长度   \dt\盒子名=长度
```

排版时, $\text{T}_{\text{E}}\text{X}$ 只按盒子的尺寸安排所占空间, 若盒中内容的实际面积大大小于盒子面积时, 就会出现很大的空白, 反之则可能会使盒中内容与外部内容重叠.

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 还提供了如下环境用于保存 LR 盒子:

```
\begin{lrbox}{\盒子名}
  文本
\end{lrbox}
```

它的功能与 `\sbox` 类似, 当保存大段的文本时, 使用这个环境较好, 以后引用时也是用命令 `\usebox{\盒子名}`. 但不要忘记, 仍要事先创建这个盒子名.

8.4.2 LR 盒子的升降

使用下述命令可以使 LR 盒子的文本竖直移位:

```
\raisebox{升高量}[盒高][盒深]{文本}
```

升高量为正时是升高, 为负时是下降. 该命令生成一个无框的 LR 盒子, 盒子的基线位于当前行的基线上, 但盒内的文本的基线相对于盒子基线竖直移动了升高量的距离.

当没有可选项 [盒高][盒深] 时, 盒子的尺寸由文本和升高量决定: 在保持盒子的基线位置不变的前提下, 自动调整盒子的大小, 使得升降后的文本刚好不会超出盒子.

当有 [盒高][盒深] 时, 盒子尺寸由给定的盒高与盒深的值确定, 此时若升高量过大, 文本会超出盒子的范围. 当盒高或盒深的值是负数时, 被当成零处理.

下面是一个例子 (见 8-4-2.tex), 为了看出盒子的范围, 给盒子加了边框:

...ABCD...EF...GH...IJ...KL...MN...

盒外的虚线点表示当前行的基线, 也是每个盒子的基线. 盒内的虚线点是盒内文本的基线. 其中

- 第一个盒子是正常的 LR 盒子 (没有升降),
- 第二个盒子是 `\raisebox{10pt}{D...E}`,
- 第三个盒子是 `\raisebox{4pt}[6pt][6pt]{H...I}`,

- 第四个盒子是 `\raisebox{-10pt}{L...M}`.



位于同一行的盒子的基线处于同一水平线上, 这些盒子组成了一个更大的盒子, 大盒子的高度与深度分别等于各个小盒子的高度和深度的最大值. 当升高或降低的盒子使用了可选项参数, 但给定的高度或深度较小而升高值较大时, 会使升高或降低的字符超出整个行盒子的范围, 从而与相邻行重叠, 这一点是需要注意的. 高度或深度为零的升降盒子对排版一些特殊的表格和矩阵是必不可少的.

8.4.3 标尺盒子

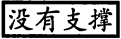
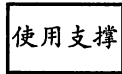
标尺盒子就是一个实心矩形, 命令

```
\rule[升高量]{水平宽度}{竖直高度}
```

生成一个具有指定宽度和高度的实心盒子. 可选项 升高量 指定了标尺盒子底部相对于当前行基线向上移动的距离, 若值是负的, 则向下移动, 若不使用这个可选项, 则标尺盒子底部就放在当前行的基线上. 下面是两个例子 (见 8-4-3.tex, 其中用细线表示基线位置):

```
\rule{5pt}{10pt}      
\rule[4pt]{20mm}{1pt} 
```

宽度为零高度不为零的盒子是看不见的, 这是一个没有宽度的柱子, 可起“支撑”作用, 比较下面两例的结果:

```
\fbox{没有支撑} 
\fbox{使用支撑\rule[-3mm]{0pt}{8mm}} 
```

8.4.4 子段盒子与小页环境

用户有两种手段可以显式地把整个段落放到一个竖直盒子中. 一种是使用子段盒子命令:

```
\parbox[位置]{宽度}{文本}
```

另一种是使用小页环境:

```
\begin{minipage}[位置]{宽度}
  文本
\end{minipage}
```

两者都生成一个具有给定宽度的竖直盒子. L^AT_EX 就像处理通常段落一样, 将盒内的文本分行, 然后堆叠成段 (这是称为“竖直”盒子的缘由). 可省略的参数位置可取如下值:

- b 盒子底行文本基线与盒子外面基线对齐
- t 盒内顶行文本基线与盒子外面基线对齐

当不使用可选项位置时,盒子的中部与盒外基线对齐.

在排版时, \TeX 对待盒子像对待单个字符一样,把盒子看成是不可分割的一个大“字符”.盒子与盒子或盒子与普通字符水平排列时,基线对齐在同一条水平线上,上下排列时,基准点对齐在同一条竖直线上.上述建立竖直盒子时的位置参数,实际上是确定竖直盒子的基线位置和基准点位置,基准点就是基线与盒子左边界的交点.

使用上述命令或环境建立竖直盒子时,没有指定总高度,生成的盒子总高度就是所含内容占据的自然总高度.

在 \LaTeX 中可以指定竖直盒子的总高度,此时还应指定盒中的内容在盒内摆放的位置,如靠上、靠下、居中或撑满等,这就需要增加两个参数,所以 \parbox 命令和 \minipage 环境更一般的形式是:

```
\parbox[位置][总高度][内部位置]{宽度}{文本}
```

```
\begin{minipage}[位置][总高度][内部位置]{宽度}
  文本
\end{minipage}
```

其中可选项内部位置可取如下值:

- t 把文本推向盒子的顶部
- b 把文本推向盒子的底部
- c 使文本竖直居中
- s 伸展行间隔使文本充满整个盒子

总高度可以是数字带一种长度单位,也可以像建立 LR 盒子时的参数一样,使用 \height 、 \depth 、 \width 和 \totalheight 参数.这些参数相当于是或不使用可选项[总高度][内部位置]时,生成的竖直盒子的相应尺寸.

应同时使用或同时不使用可选项[总高度][内部位置],以避免意外错误.

使用子段盒子或小页环境可实现图文混排如第 81 页的示例.

§ 8.5 制表位的高级技巧

制表位从左到右顺序编号,最左边是零号制表位.在 \tabbing 环境中,每一行都是默认从零号制表位开始.但若把 \+ 放在一行的开头或末尾,那么后继行的开始位置就都右移一个制表位.如果用 $\text{\+}\text{\+}$ 开始或结束一行,后继行开始位置就都右移两个制表位,依此类推.一行设置过几个制表位,前面最多就可以使用几个 \+ .

命令 \- 具有相反的效果, 每用一次 \-, 就抵消一个 \+ 的作用, 也就是说, 当用 \- 开始或结束一行时, 后继行的开始位置就都向左移动一个制表位. 注意在任何位置 \- 的总数不能超过 \+ 的总数.

一行上所有制表位的位置构成一个“制表位表”. 为了清除制表位表以便重新设置制表位, 可以在一行开始时使用命令 \pushtabs, 它将现有的制表位表保存到堆栈中以备后用, 然后清除所有的制表位, 于是就可以重新设置制表位了. 在一行开始使用命令 \poptabs, 它也是清除当前的各个制表位, 但同时把最近保存在堆栈中的制表位表取出来, 恢复表中的各个制表位. 可以多次使用 \pushtabs, 保存不同行上的制表位表. 也可以多次分别地或连续地使用 \poptabs, 恢复以前某一次保存的制表位表.

注意, 在任何时候, \poptabs 的数目都不能超过已有的 \pushtabs 的数目, 并且在 tabbing 环境结束时, 要保证 \poptabs 与 \pushtabs 的数目必须相同. 此外, \poptabs 并不保存或取消 \+ 和 \-, 因此不影响任何已有的 \+ 和 \- 的作用.

通常情况下, 在 tabbing 环境中, 位于相邻制表位之间的文本, 都是左端对齐在左边一个制表位上 (下称当前制表位). 如果输入的文本形如

左边文本 \ 右边文本

则是 \ 对应的位置对齐在当前制表位上 (该符号本身不显示), 严格讲是 右边文本左对齐在当前制表位处, 左边文本右对齐当前制表位, 但左边文本与制表位之间还有一个小小的间隔, 其值由参数 \tabbingsep 确定. 用户可用命令 \setlength 改变这个参数的值. 可以看出 左边文本 实际上是位于前面一列的区域内, 因此要当心不要与前一列文本重叠.

命令

\ 文本

使文本右对齐于当前版面的右边界. 在文本与该行换行命令 \\ 之间不能再出现 \> 或 \= 命令.

在 \tabbing 环境外面, \=、\' 和 \` 都是重音符号, 如果在该环境内需要这些重音符号, 就要把 \ 改为 \a. 例如若在 tabbing 环境生成字符 ó、ò 或 ô, 就必须输入成 \a'o、\a'o 或 \a=ó. 命令 \- 在 tabbing 环境之外是建议断词标志, 但该环境中不允许断行 (只能用 \\ 换行), 所以就不需要为它定义替代形式了.

§8.6 表格的高级技巧

8.6.1 更多的表格参数

在基础篇中已对表格的常用参数作了介绍 (参见第 37 页), 这里再补充介绍一些更深入的参数.

构造表格的环境有两个:

```
\begin{tabular}[竖直位置]{列格式}
.....
\end{tabular}
```

```
\begin{tabular*}{整表宽度}[竖直位置]{列格式}
.....
\end{tabular*}
```

其中带*号的环境指定了表格总宽度,为达到这个宽度必须在列的间隔中含有弹性长度,为此可在列格式的某些地方插入命令`@{\extracolsep\fill}`(见下面介绍),以使后面的列间距可以伸展,保证表格能达到指定的宽度.对于不带*号的环境,表格宽度取决于各列内容的宽度.

在基础篇中已经介绍过竖直位置参数`t`与`b`,以及列格式参数`l`,`c`,`r`,`|`,`||`,下面再补充一些列格式的参数:

`p{宽度}` 指定对应的列的宽度,列的内容过多时会自动分行,顶行与其他列对齐.相当于使用了命令`\parbox[t]{宽度}{列文本}`处理该列文本;

`*{数}{格式}` 这是几个相同格式的缩写,例如可将格式串`|r|l|r|l|r|l|`缩写成`|*{3}{r|l}|`或`*{3}{|r|l}|`.

对应于边界或列间距的格式项还有:

`@{文本}` 称为@表达式,在它对应的位置(表格的边界或是两列之间)“吃掉”了原有的列间隔空白,改为插入指定的文本.

若无文本,只写`@{}`,就相当于取消了对应位置的列间隔.在表格的边界处,若不画竖线,则边界处就有宽度为列间距一半的空白.若想删除这种空白,可在列格式的开始和结束处写上格式项`@{}`.

如果在列之间插入的指定文本与相邻列之间需要一些空白,那么必须在@表达式的文本中包含设置空白的命令,如`\hspace{...}`.如果不想在两列之间插入文本,只想改变间距使其与默认间距不同,只要写成`@{\hspace{宽度}}`就行了,参数值可以是负数.

如果在@表达式中含有`\extracolsep{宽度}`,则会使其后所有的列间距都增加指定的额外宽度,直到遇到下一个同样命令为止.这种额外的间距不会被后面的@表达式吃掉.

控制表格样式的参数有:

<code>\tabcolsep</code>	<code>tabular</code> 和 <code>tabular*</code> 表格环境列间距的一半
<code>\arraycolsep</code>	<code>array</code> 表格环境列间距的一半

`\arrayrulewidth` 各类表格中画线 (包括横线和竖线) 的粗细
`\doublerulesep` 双线之间的距离
`\arraystretch` 各类表格中的行距伸缩因子

这些参数都有预定值, 符合大多数人的要求. 用户有特殊要求时, 可以改变预定值. 例如

```
\setlength{\tabcolsep}{2pt}
\renewcommand{\arraystretch}{1.2}
```

设置了两列之间的间距为 4pt, 将表格行的行距增大了 20%.
上面提到的 array 表格环境通常称为阵列环境, 格式为

```
\begin{array}[竖直位置]{列格式}
  第一行\\
  第二行\\
  ..... \\
  第末行
\end{array}
```

其中参数的意义与 tabular 的相同, 只是这个环境只能用于数学模式, 大多用于排版矩阵或方程组, 详见第四章.

8.6.2 几个表格样例

例1 下面是一个行高和“列数”都有变化的表格:

Font Size		
Huge 25	huge 20	
LARGE 17	Large 14	large 12
normalsize 10	small 9	footnotesize 8
scriptsize 7	tiny 5	

这个表格是如下输入的 (参见 8-6-1.tex):

```
\newcommand{\ZZ}[2]{\rule[#1]{0pt}{#2}}
\newcommand{\MC}[3]{\multicolumn{#1}{#2}{#3}}
\begin{center}\begin{tabular}{|c|c|c|}
  \MC{3}{c}{\textbf{Font Size}}\|[5pt] \hline
  \MC{3}{|c|}{\ZZ{-8pt}{30pt}\hfill\Huge Huge 25\hfill}
  \vline\hfill\huge huge 20\hfill\mbox{}}\|[ \hline
```

```

\ZZ{-6pt}{22pt}\LARGE LARGE 17 & \Large Large 14
& \large large 12\\ \hline
\normalsize normalsize 10 & \small small 9
& \footnotesize footnotesize 8\\ \hline
\scriptsize scriptsize 7 & \tiny tiny 5 &\\ \hline
\end{tabular}\end{center}

```

在输入这个表格之前定义了两个新的命令,这主要是为了简化输入工作.对于频繁使用的较长命令,定义一个较短的命令,不仅减少了输入量,也减少了输入出错的机会.对于这个例子,要特别注意3点:

第一,为了使标题与表格固连在一起,永远不被分在两页,可以把标题当成是表格的一行,但不使用表格的列格式,这时就要用`\multicolumn`命令了.此外还用了`\\[5pt]`命令增加与下面行的间隔.

第二,有最大文字的那行看起来只有两列,也不符合本表格的列格式,所以也要用`\multicolumn`命令.该行还用了`\vline`打印一段竖线,并使用了可以无限伸长的弹性长度`\hfill`,使竖线两边的文字具有居中的效果.

第三,在有很大文字的行使用了“看不见的支柱”——新定义的`\ZZ`命令,以撑高表格高度,使表格线与文字之间有一定的间隔.

例2 下面表格(见8-6-2.tex)用到了列格式参数`p{宽度}`和命令`\cline`与`\raisebox`.另外为了简化输入,又定义了几个命令.

姓名		性别		贴照片处
生日	□□□□年□□月□□日			
住址				
通讯	电话:			
	E-mail:			
备注				

这个表格是如下输入的:

```

{
\newsavebox{\fk}\newsavebox{\kk}
\setlength{\fboxsep}{0pt}
\setlength{\fboxrule}{0.3pt}
\sbox{\fk}{\framebox[3mm]{\rule[-2pt]{0pt}{3mm}}}
\sbox{\kk}{\usebox{\fk}\usebox{\fk}}
\begin{center}
\begin{tabular}{|c|l|p{20mm}|}
\hline
姓名 & \hspace{\stretch{3.5}}%
& \vline\,\,\,性别\,\,\,\vline\hspace*{\fill} & \\

```

```

\cline{1-2}
生日 & \usebox{\kk}\usebox{\kk}\,\,\,年\,\,\,%
      \usebox{\kk}\,\,\,月\,\,\,\usebox{\kk}\,\,\,日 & \\
\cline{1-2}
住址 & & \hfill贴照片处\hfill{} \\
\cline{1-2}
      & 电话: & \\
\cline{2-2}
\raisebox{1.6ex}[Opt]{通讯} & E-mail: & \\
\hline
备注 & \MC{2}{c|}{c|}{c|} \\
\hline
\end{tabular}
\end{center}
}

```

注意输入表中“通讯”两字时,使用了上升的盒子,抬高了这两个字的位置. 这里升高盒子的可选项 [Opt] 决不可省, 这是因为该项表示盒子高度为零, 盒内的文本虽然升高了, 但因盒高为零, 所以不影响当前行的高度. 如果省略了这个盒高选项, 那么随着盒内文本的升高, 当前行的高度也随着增大, 就与其他行的高度不同了. 想一想, 如果将“贴照片处”放在“生日”行输入, 但仍要显示在上下居中的位置, 应如何输入? 注意, 如果设置升降盒子的深度为零, 不能只写一个可选项 [Opt], 这是因为当只有一个可选项时, 默认它代表盒子高度而不是深度.

在表格第二列中, 第二行中的内容最长, 它决定了第二列的宽度. 第一行第二列只有两个汉字和两条竖线, 所以该单元格有很多的空白. 为了让空白按 3.5:1 的比例分布在文字两侧, 在“| 性别 |”的右边用了一个不会被吃掉的 \hfill “弹簧”, 在左边则放了三个半“弹簧”. 命令

```
\stretch{倍数}
```

生成一个弹性长度, 是 \fill 的指定倍数, \fill 的基本长度为 0, 可以无限伸展到任意需要的长度.

本例的输入内容被放在一个分组内, 即用花括号括了起来, 这是为了使改变后的参数值 \fboxsep 和 \fboxrule 仅在这个组内起作用.

例3 下面表格元素在小数点处对齐, 使用了 @ 表达式, 如下输入(见 8-6-3.tex):

```

\begin{center}
\begin{tabular}{cr@{.}l}
\multicolumn{3}{l}{数学表达式\quad 值\,(10\,位数字)} \\
\hline\hline
$\pi$ & 3 & & 141592654 \\

```

```

 $\pi^2$  & 9 & 869604401\\
 $\pi^3$  & 31 & 00627668\\
 $\pi^{10}$  & 93648 & 04748\\
 $\pi^{20}$  & 8769956796\\
 $1/\pi$  & 0 & 3183098862
\end{tabular}
\end{center}

```

结果为

数学表达式	值 (10 位数字)
π	3.141592654
π^2	9.869604401
π^3	31.00627668
π^{10}	93648.04748
π^{20}	8769956796.
$1/\pi$	0.3183098862

§ 8.7 罗列

8.7.1 三种罗列环境

写文章做报告时, 常常要将文字材料分成几个层次, 按甲乙丙丁一二三四罗列出来. \LaTeX 提供了三种罗列环境:

```

\begin{itemize}
  罗列条目1
  罗列条目2
  ...
  罗列条目n
\end{itemize}

```

```

\begin{enumerate}
  罗列条目1
  罗列条目2
  ...
  罗列条目n
\end{enumerate}

```

```
\begin{description}
  罗列条目1
  罗列条目2
  ...
  罗列条目n
\end{description}
```

其中所有 罗列条目 都有如下形式:

```
\item[标签] 条目文本
```

输出时在每个条目的前面显示 标签, 标签既可以是符号, 也可以是文字. 若一个条目文本的内容很长, 会自动分行, 分出来的后面几行与第一行的文字对齐, 标签则凸出悬挂于条目的第一行左端, 其中环境 `description` 还会用黑体显示标签.

上述 3 种环境的区别是对可省略的参数 “[标签]” 的处理方式不同.

当不写 [标签] 时, 环境 `itemize` 会自动生成默认的条目标签, 其特点是并列的条目有同样的标签; 环境 `enumerate` 也会自动生成默认的条目标签, 那是自动排序编号的符号; 但环境 `description` 没有默认的标签, 因此在这种罗列环境中不要省略 [标签] 项.

上述 3 种罗列环境可以彼此相互嵌套, 每一种环境最多嵌套 4 层, 嵌套时内层的环境会向右整体缩进. 前两种环境的默认条目标签与该环境自己的嵌套层数有关. `itemize` 环境的 4 层默认标签分别是 \bullet , $-$, $*$ 和 \cdot . `enumerate` 环境第一层默认标签是阿拉伯数字后跟一个圆点句号, 第二层是圆括号包围的小写拉丁字母, 第三层是小写罗马数字后跟圆点句号, 第四层是大写拉丁字母后跟圆点句号, 标签是自动顺序编号的, 如果某个罗列条目指定了标签, 则该项不参加自动编号.

8.7.2 改变默认的罗列条目标签

改变某一项的默认标签是很简单的, 只要在输入 `\item[标签] 条目文本` 时, 不省略标签就行了. 若要改变所有项的默认标签, 需先了解 L^AT_EX 的一些内部命令.

`itemize` 环境的 4 层默认标签分别保存在下列 4 个命令中:

```
\labelitemi      \labelitemii     \labelitemiii    \labelitemiv
```

容易猜出命令名字中的 `i`, `ii`, `iii`, `iv` 分别指的是四个层次中的一层. 可以用重定义命令改变每层的默认标签, 例如将第二层的默认标签 “ $-$ ” 改为 “ $+$ ”, 可使用命令:

```
\renewcommand{\labelitemii}{+}
```

类似地, 在 `enumerate` 环境中保存每层标签的命令是

```
\labelenumi      \labelenumii     \labelenumiii    \labelenumiv
```

但因标签是顺次编号的, 所以对应于每层还有一个计数器, 计数器名字分别是 (没有倒斜线!):

```
enumi    enumii    enumiii    enumiv
```

计数器的值可以用下列 5 种命令显示成不同形式:

```
\arabic    \roman    \Roman    \alph    \Alph
```

从名字上就可知道它们分别表示 阿拉伯数字、小写罗马数字、大写罗马数字、小写拉丁字母和大写拉丁字母。

如果要把 `enumerate` 环境第一层的标签改为大写字母 A、B、C 等, 把第二层标签改为阿拉伯数字, 但还要显示出它所在的上一层的序号, 即显示的标签形如 A.1、A.2、B.1 等, 第三层不变, 第四层是用小于号和大于号括起来的小写罗马数字, 可如下使用重定义命令:

```
\renewcommand{\labelenumi}{\Alph{enumi}}
\renewcommand{\labelenumii}{\Alph{enumi}.\arabic{enumii}}
\renewcommand{\labelenumiv}{\$<\roman{enumiv}\$>}
```

如果把上述重定义命令放在导言区, 则新的默认标签对整个文档起作用, 否则就只在它们所处的环境或分组内起作用。

下面是一个嵌套例子 (见 8-7-1.tex), 没有重定义默认标签, 输入的原稿如下。输入时使用了缩排格式, 便于检查错误。

```
\begin{itemize}
  \item 这是~itemize~环境的第一层, 标签是黑圆点.
  \begin{enumerate}
    \item 虽然是总的第二层, 但这是~enumerate~环境的第一层.
    \begin{enumerate}
      \item 这是~enumerate~环境的第二层.
      \item 条目标签与总的嵌套层数无关,
            只取决于它在所属的罗列环境中的层数.
    \end{enumerate}
    \item 回到了~enumerate~环境第一层, 是这个环境的第二个条目.
  \end{enumerate}
  \item 回到了~itemize~环境, 条目标签又变成了黑圆点.
\end{itemize}
```

输出结果如下:

- 这是 itemize 环境的第一层, 标签是黑圆点.

1. 虽然是总的第二层, 但这是 `enumerate` 环境的第一层.
 - (a) 这是 `enumerate` 环境的第二层.
 - (b) 条目标签与总的嵌套层数无关, 只取决于它在所属的罗列环境中的层数.
 2. 回到了 `enumerate` 环境第一层, 是这个环境的第二个条目.
- 回到了 `itemize` 环境, 条目标签又变成了黑圆点.

§ 8.8 广义罗列环境

上述三种罗列环境以及其他一些列表都可以通过一个非常一般的环境构造出来. 标签的类型与宽度, 缩进的距离, 项目、段落、标签等相互之间的距离都可以整体或部分的调整和设置. 这就是广义罗列环境 `list`:

```
\begin{list}{标准标签}{罗列声明}
\item[标签] ...
:
\item[标签] ...
\end{list}
```

罗列的每一项都由 `\item` 引导以便生成相应的标签. 若 `\item` 不带 [标签] 可选项, 则使用 标准标签 的内容作为该项的标签.

有很多参数控制广义罗列环境的排版效果, 用户可以使用默认值, 也可在 罗列声明 中将参数设置为所期望的值.

8.8.1 标准标签

标准标签 就是默认标签, 当广义罗列环境不带可选项 [标签] 时, 就使用 标准标签 作为它的标签.

如果希望 标准标签 是自动递增的数字, 就必须使用命令

```
\newcounter{名称}
```

建立一个新的计数器, 该命令应放在第一次使用它的计数值之前. 与其他计数器相同, 可以用不同的形式显示计数器的值 (见第 100 页).

若在 标准标签 中使用一个计数器, 必须在 罗列声明 中包含命令

```
\usecounter{名称}
```

8.8.2 广义罗列环境的样式参数

有很多样式参数控制广义罗列环境的排版效果, 它们都有默认值, 用户可以在罗列声明中用`\setlength`命令改变这些值. 由于每个参数在每一层都有预设的默认值, 而这些值只有在罗列声明中才可以被覆盖, 所以在广义罗列环境之外通常不能重设参数的值.

下面介绍广义罗列环境的样式参数, 需参考图 8.1 以便于理解.

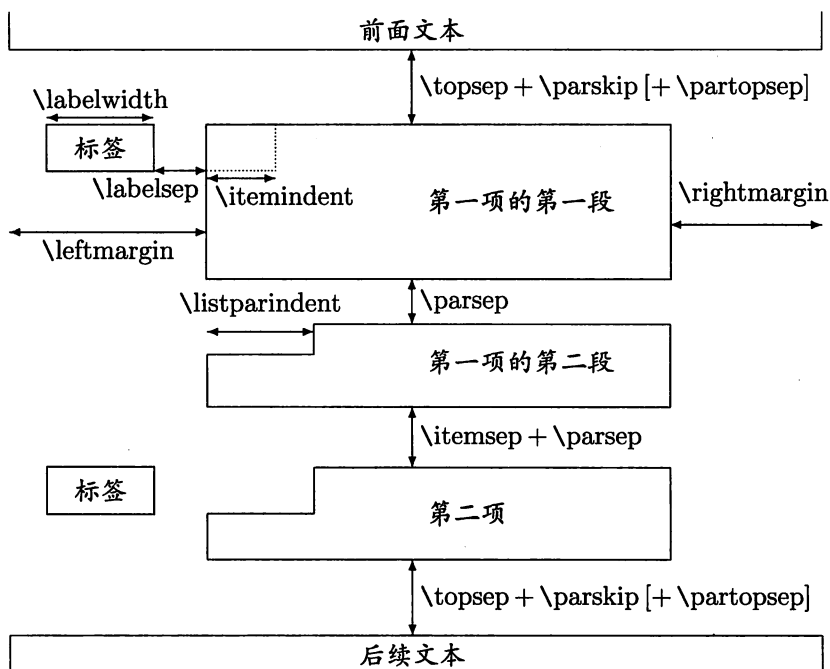


图 8.1 广义罗列环境参数示意图

`\topsep` 是一个竖直间距, 与 `\parskip` 加在一起插入到罗列环境与包围它的外部文本 (图中显示的前面文本和后续文本) 之间. 其值需在每个罗列环境中设置, 即使罗列环境嵌套, 也需在每个环境的罗列声明中改变默认值, 不能在罗列环境之外作全局设定.

`\partopsep` 当在广义罗列环境的前面或后面有空行时, 就会把这个参数的值与上一条提到的两个参数的值加起来, 插入到该环境与外部的文本行之间.

`\parsep` 相当于普通文本的 `\parskip` 参数, 但用在广义罗列环境内, 控制在同一个 `\item` 项目中, 各个段落之间的竖直间距. 需在每个环境的罗列声明中改变默认值, 不能在罗列环境之外作全局设定.

`\itemsep` 是一个竖直间距, 与 `\parsep` 加在一起插入到相邻两个 `\item` 项目之间.

`\itemindent` 是标签盒子与条目文本第一行向右缩进的距离. 通常设为 0pt, 因此没有这种缩进. 只能在罗列声明中改变其值. 图中用虚线表示了文本缩进的这个距离, 没有反映出标签也应缩进同样的距离. 通常罗列的每一项都是悬挂效果(首行突出), 如果要改变成首行缩进(连标签一起缩进), 就需要使用这个参数了.

`\listparindent` 相当于普通文本的 `\parindent` 参数, 是 `\item` 项中每一个段落的第一行相对于条目文本段落左边界缩进的距离. 默认值为 0pt, 只能在罗列声明中改变.

`\labelwidth` 是为标签保留的盒子的宽度. 在这个盒子中, 标签靠右对齐在这个盒子的右边线上. 但若标签的长度超过了 `\labelwidth` 的值, 这个盒子会向右伸展以拟合标签的长度, 此时条目文本段落的首行将向右缩进, 在标签与条目文本首行之间仍有 `\labelsep` 的间隔. 可以为这个参数设置一个全局性的值以适用于所有的罗列层次.

`\labelsep` 是标签盒子与条目文本段落左边界之间的距离. 可以在外部为其设置一个全局性的值, 但只对嵌套罗列的第一层起作用.

`\leftmargin` 是当前罗列环境内条目文本段落的左边界与外部环境左边界之间的距离.

`\rightmargin` 是当前罗列环境内条目文本段落的右边界与外部环境右边界之间的距离. 默认值为 0pt, 只能在罗列声明中改变.

容易看出, `\labelwidth+\labelsep` 是标签盒子左边界与条目文本段落左边界之间的距离. `\leftmargin-(\labelwidth+\labelsep)` 就是标签盒子左边界与外部环境左边界之间的距离, 当这个值为负数时, 标签盒子将向左突出外部环境的左边界.

利用上述参数, 可以随心所欲地设计各式罗列(列表)格式, 例如想要罗列的各项之间以及同一项内的各段之间的间距都与正常文本的段落间距相同, 各项标签左对齐, 标签含有序号, 左右相对于正文缩进 2em, 形如

问题 9: A、B、C 三射手各自独立地向同一目标进行一次射击, 已知 A、B、C 射中目标的概率分别为 $\frac{1}{3}$ 、 $\frac{1}{2}$ 、 $\frac{1}{4}$, 则至少有一人射中目标的概率为 _____.

问题 10: 设 $\xi \sim N(\mu, \sigma^2)$, 则 $D\xi =$ _____.

可以如下输入(见 8-8-1.tex):

```
\newcommand{\TKX}[1]{\,\,\,\,\,\rule[-3pt]{#1mm}{0.5pt}}
\newcounter{timu}
\begin{list}
{\bfseries\sffamily 问题\,\,\arabic{timu}:\hfill}
```

```

{\setlength{\parsep}{\parskip}
 \setlength{\itemsep}{0ex plus0.1ex}
 \setlength{\labelwidth}{4em}
 \setlength{\labelsep}{0.2em}
 \setlength{\leftmargin}{6.2em}
 \setlength{\rightmargin}{2em}
 \usecounter{timu} \setcounter{timu}{8}
 \upshape
}
\item A、B、C 三射手各自独立地...概率为\TKX{12}.
\item 设..., 则 $\xi=\text{\TKX{12}}$ .
\end{list}

```

根据罗列参数的值, 容易算出 $6.2\text{em} - 4\text{em} - 0.2\text{em} = 2\text{em}$ 就是标签左边界与页面左边界的距离。

注意语句 `\usecounter{timu}\setcounter{timu}{8}` 的顺序, 如果次序颠倒, 则第一个标签序号是 1。

在标准标签和罗列声明中可以分别指定字体属性。

8.8.3 平凡罗列环境

在 \LaTeX 中还有一个平凡罗列环境:

```
\begin{trivlist} 文本 \end{trivlist}
```

它相当于一个广义罗列环境, 但没有标准标签和罗列声明参数, 其标签是空的, `\leftmargin`、`\labelwidth` 和 `\itemindent` 都赋值为 0pt, 而 `\listparindent` 等于 `\parindent`, `\parsep` 等于 `\parskip`。

\LaTeX 用这一环境生成其他一些环境。例如当调用 `center` 环境时, 内部实际上是调用

```

\begin{trivlist}
\centering \item[] 文本
\end{trivlist}

```

环境 `flushleft` 和 `flushright` 也是类似定义的。

§ 8.9 脚注与边注

写在页面底部的注释称为脚注, 写在页面边上的注释称为边注。关于脚注的基本知识请参见基础篇的第 40 页。

8.9.1 自动编号的脚注

L^AT_EX 内部有一个脚注计数器 `footnote`, 每次调用 `footnote`, 计数器的值就加 1, 并用阿拉伯数字显示计数器的新值作为脚注标记. 可以使用命令

```
\setcounter{footnote}{初值}
```

重设计数器的初值, 以后的第一个脚注编号等于初值 +1. 可以用如下命令改变脚注的数字形式

```
\renewcommand{\thefootnote}{\数字样式{footnote}}
```

其中数字样式就是在第 100 页介绍的 6 种计数器数字样式命令. 对于计数器 `footnote`, 最常用的是使用计数器数字样式命令 `\fnsymbol`, 它把从 1 到 9 的计数器值显示为下列 9 个符号:

```
* † ‡ § ¶ || ** †† ‡‡
```

由于只有 9 个符号, 所以在调用第十个脚注之前应重设计数器值为 0.

使用下述命令可把脚注标记恢复成阿拉伯数字形式:

```
\renewcommand{\thefootnote}{\arabic{footnote}}
```

在小页环境中, 有独立的脚注计数器, 名为 `mpfootnote`, 与计数器 `footnote` 无关. 小页中脚注的默认标记是小写拉丁字母.

8.9.2 指定编号的脚注

下述命令直接指定脚注编号, 并且不改变脚注计数器的值:

```
\footnote[数]{脚注文本}
```

其中数是一个整数. 如果指定了脚注计数器的显示形式为 `\fnsymbol`, 则数应位于 1 到 9 之间.

8.9.3 禁止模式中的脚注

前面说过脚注命令不能位于数学模式、表格、LR 盒子或子段盒子中, 我们把这些模式称为禁止脚注模式. 但可以把脚注命令分成两部分, 在需要注释的地方只写脚注标记, 这可以出现在任何地方, 包括出现在禁止模式内部, 而把脚注文本放在禁止模式之外. 分别写脚注标记和脚注文本的命令为

```
\footnotemark  
\footnotetext{脚注文本}
```

或

```
\footnotemark[数]
\footnotetext[数]{脚注文本}
```

第一组命令对应于自动编号的脚注命令, 第二组对应于指定编号的脚注命令. 每组的第一条命令(脚注标记命令)总是紧接在需要注释的文字后面, 排版输出时就在它出现的地方显示脚注标记. 第二条命令(脚注文本命令)则放在禁止模式的外面. 对于同一个脚注, 必须使用同一组的脚注标记命令和脚注文本命令.

需要注意的是, 每遇到一个自动编号的脚注标记命令, 脚注计数器的值就自动加1, 而自动编号的脚注文本命令只使用计数器的值, 不改变计数器的值. 当在禁止模式中使用了多个自动编号的脚注标记命令, 则脚注计数器的值就增加了几次. 等到了禁止模式, 遇到的第一个自动编号的脚注文本命令使用的计数器的值就与第一个自动编号的脚注标记的值不同了. 解决办法是在第一个自动编号的脚注文本命令之前调整脚注计数器的值:

```
\addtocounter{footnote}{负数}
```

这里 负数 的绝对值应是自动编号的脚注标记命令的个数减1. 在第二个以及以后的每个自动编号的脚注文本命令之前都要使用下述两个命令之一给脚注计数器加1:

```
\addtocounter{footnote}{1}
\stepcounter{footnote}
```

8.9.4 边注

在页边上的注释可以使用下列命令生成:

```
\marginpar{边注文本}
```

它把边注文本显示在当前页面的右侧页边上, 边注第一行基线与命令所在行基线齐平. 边注文本通常被放在一个较窄的子段盒子中, 因此要对边注文本分行.

默认边注出现在页面右边. 如果使用了双面选项 `twoside`, 则显示在“外”边, 即奇数页显示在右页边, 偶数页显示在左页边. 若使用了双列选项 `twocolumn`, 则显示在两列的“外”侧, 即左列的左侧, 右列的右侧. 本页出现的边注是插入命令 `\marginpar{\small这是一个边注}` 得到的.

`\marginpar` 把边注文本作左对齐处理, 这当边注位于页面右边时没有问题, 当位于左边时就会离正文过远, 此时应在边注文本前加上命令 `\flushright`.

如果边注带有方向, 例如是一个指向页芯的箭头, 那么位于左边或右边时, 边注的内容应有所不同, 此时应使用命令

```
\marginpar[\flushright左边注文本]{右边注文本}
```

对于 book 文档类, 默认使用 `twoside` 选项, 但准备文稿时不知道边注将来位于哪一页, 所以应使用上述命令, 其中左边注文和右边注文可以是相同内容. 本书的边注实际就是这样输入的.

如果需要改变边注的默认位置, 使其出现在页面的另一侧, 则应事先使用命令

```
\reversemarginpar
```

控制边注样式的参数有

<code>\marginparwidth</code>	边注盒子的宽度
<code>\marginparsep</code>	边注盒子与正文边界之间的距离
<code>\marginparpush</code>	两个边注盒子之间最小竖直距离

这些参数都是长度, 可用 `\setlength` 命令赋予一个新值.

§ 8.10 段落形状

通常情况下段落的形状是首行缩进, 但有时需要首行突出悬挂. 段落的边界通常对齐在页芯的两边, 但有时需要段落边界整体向内缩进或整体向外突出, 或者段落的某些行缩进或突出. 使用 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 原有的一些命令可以容易地改变段落形状. 虽然不提倡在 $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 中直接使用 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 原有的命令, 但是当遇到一些特殊问题时, 若 $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 没有简便方法, 而 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 原有命令可以方便地解决问题时, 就应该直接使用 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 命令以简化操作. 需要注意 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 的命令格式不同于 $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 的风格, 一般带参数的格式是 `\命令=参数`, 其中等号可以省略.

8.10.1 移动段落边界

使用 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 命令

```
\leftskip=长度值 \rightskip=长度值
```

可以分别使段落左、右边界整体移动, 正的长度值使边界向内缩进, 负的长度值则使边界向外突出. 在本段放置了命令

```
\leftskip=30mm\rightskip=-5mm
```

所以才有现在的效果. 为了不影响后面的段落, 应在本段落结束后将这两个命令重新赋值为 `0mm`. 若一段中有几个相同的这种命令, 则最后一个起作用.

8.10.2 多行缩进

TeX 命令

<code>\hangafter=整数</code> <code>\hangindent=长度</code>
--

分别控制缩进的行数与距离。只对当前段落起作用。需注意`\parindent`与这两个命令无关,总是控制首行缩进(负值则首行突出)。必要时在段首放置命令`\noindent`以取消其影响。

- `\hangafter ≥ 0`: 段落开始几行不受影响,行数等于命令给定的数字。其后各行才缩进,达到了多行悬挂的视觉效果。该命令的默认值是1。

- `\hangafter < 0`: 段落开始几行缩进,行数等于给定数字的绝对值。其后各行不受影响。

- `\hangindent ≥ 0`: 其值指定了受`\hangafter`影响的各行左端缩进的距离。该命令的默认值是0pt。

- `\hangindent < 0`: 给定值的绝对值指定了受`\hangafter`影响的各行右端缩进的距离。

上述命令4种组合的效果如下(行数值为±2, 缩进值为±3em):

<pre> a a a a a a a a a a a a a a a a a a a </pre>	<pre> \hangafter=2\hangindent=3em\noindent a a a a a a a a a ... </pre>
<pre> a </pre>	<pre> \hangafter=2\hangindent=-3em\noindent a a a a a a a a a ... </pre>
<pre> a a a a a a a a a a </pre>	<pre> \hangafter=-2\hangindent=3em\noindent a a a a a a a a a ... </pre>
<pre> a </pre>	<pre> \hangafter=-2\hangindent=-3em\noindent a a a a a a a a a ... </pre>

8.10.3 段落形状命令

TeX 命令

<code>\parshape = n i₁ l₁ i₂ l₂ ... i_n l_n</code>
--

可以控制段落每行的缩进值与长度, 其中等号可省略. 参数 n 是一个非负整数, 表示该命令控制的行数; 参数 i_1, i_2, \dots, i_n 分别表示各行左端缩进的距离 (负值则是突出的距离); 参数 $\ell_1, \ell_2, \dots, \ell_n$ 分别表示各行的长度. 该命令只对当前段落起作用, 若本段排版后不足 n 行, 则多余的 i, ℓ 参数被忽略. 若行数超过 n 行, 则重复最后一对参数. 此外要注意参数的总个数是 $2n + 1$ 个, 不要搞错. 在 Knuth 的名著 *The T_EXbook* 一书中 (第 101 页), 有一个使用 `\parshape` 命令的例子, 我们把它稍作简化, 并把几个命令改用了 L^AT_EX 命令, 放在文件 8-10-1.tex 中, 可以编译显示出来欣赏, 注意该文件中使用的字体为 10 pt, 如果改用 11 pt 或者 12 pt 字体, 需重新计算调整 `\parshape` 命令的参数. 本段排版使用的 `\parshape` 各参数值如下所示:

```
\parshape=13
0mm    120mm
5mm     110mm
10mm    100mm
15mm     90mm
20mm     80mm
25mm     70mm
30mm     60mm
25mm     70mm
20mm     80mm
15mm     90mm
10mm    100mm
5mm     110mm
0mm     120mm
```

第九章 数学公式排版的一些技巧

利用第四章的知识, 已经可以比较方便地排版普通的数学公式, 但要排版一些复杂的公式, 还是很麻烦的事情. 本章的前半部分介绍一些高级的数学排版知识和技巧, 后半部分介绍宏包 `amsmath`, 它提供了很多命令, 大大增强了复杂公式的排版功能.

§ 9.1 数学排版的国际标准

国际标准组织 (ISO) 已经建立了数学排版的一套公认标准. 其中值得注意的有以下几条:

1. 用粗斜体表示向量, 例如: $\boldsymbol{B} \boldsymbol{v} \boldsymbol{\omega}$;
2. 用无衬线斜体表示矩阵与 2 阶张量, 例如: $\boldsymbol{M} \boldsymbol{D} \boldsymbol{I}$;
3. 特殊常数如 e , i , π , 以及微分算子 d , 都要采用直立体, 以与变量区分;
4. 由数与单位共同构成的度量应看作一个不可分割的整体, 在两者之间有一个小的间隙, 而且单位要用直立体表示, 例如: 5.3 km , 62 kg 等.

其中第 4 条很容易用命令 `\`, 实现, 例如: 输入 `1.80\,m` 就可得到 1.80 m .

对于第 1 条, 如果我们已经调入了宏包 `amsbsy` 或 `bm`, 则可用以下的命令之一重新定义 `\vec`:

<code>\renewcommand{\vec}[1]{\boldsymbol{#1}}</code>	(宏包 <code>amsbsy</code>)
<code>\renewcommand{\vec}[1]{\bm{#1}}</code>	(宏包 <code>bm</code>)

否则, 可使用以下命令:

<code>\renewcommand{\vec}[1]{\mbox{\boldmath\$#1\$}}</code>

这样就可输入 `\vec{a}` 得到 \boldsymbol{a} .

对于第 3 条, 我们可以分别定义 3 个命令 `\me`, `\mi` 以及 `\dif` 以在数学公式内生成常数 e , i 及微分算子 d :


```
\newcommand{\me}{\mathrm{e}}
\newcommand{\mi}{\mathrm{i}}
\newcommand{\dif}{\mathrm{d}}
```

关于直立体 π 的打印问题, 可参见第 63 页.

为了达到第 2 条的要求, 我们必须定义一种新的数学字体, 名为 `\mathsfsl`, 再利用这个字体定义一个名为 `\tensor` 的命令 (注意前面一条命令必须出现在导言中):

```
\DeclareMathAlphabet{\mathsfsl}{OT1}{cmss}{m}{sl}
\newcommand{\tensor}[1]{\mathsfsl{#1}}
```

这样我们只要输入 `$\tensor A$` 就能得到 A .

对于希望能用 ISO 国际标准打印自己的文稿的读者, 不妨把上面的命令做成一个文件, 取名为 `isomath.tex`, 然后在 $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 源文件的头部用命令 `\input{isomath}` 调入此文件.

§ 9.2 数学模式中的字体尺寸

在数学模式中, 有 4 种控制字体相对大小的字体尺寸. 为行文方便, 在本章中用一个或两个大写字母表示它们:

<code>\displaystyle</code>	D	行间公式的基本尺寸
<code>\textstyle</code>	T	行内公式的基本尺寸
<code>\scriptstyle</code>	S	一级角标的尺寸
<code>\scriptscriptstyle</code>	SS	二级角标的尺寸

对于一个变量如 x , 用 D 尺寸或 T 尺寸显示出来是一样大的.

进入数学模式后, 立即被激活的字体尺寸是 D (行间公式) 或 T (行内公式). 以这两种尺寸为基础, 不同的数学要素使用不同的其他尺寸, 例如当前是 D 尺寸时, 分数使用 T 尺寸, 如果当前是 T 尺寸, 则分数使用 S 尺寸.

四种数学字体尺寸各自对应有一个修正版本, 为方便将修正版本加撇表示. 差别是 D' , T' , S' , SS' 的上标要比对应的 D, T, S, SS 的上标稍稍下降了一点点. 在其他情况, 有撇和无撇的字体尺寸是相同的. 不同数学要素选择字体尺寸的规则见表 9.1.

有些符号具有两种尺寸, 当前为 D 时, 显示为大符号, 当前为 T 时, 显示为小符号.

在 `array` 环境中, 激活的字体尺寸是 T.

当在分子或上标中显式地指定字体尺寸时, 实际使用的是无撇形式; 而在分母或下标中实际使用的是修正版本.

表 9.1 数学字体尺寸规则

激活尺寸	分子	分母	上标	下标
D	T	T'	S	S'
D'	T'	T'	S'	S'
T	S	S'	S	S'
T'	S'	S'	S'	S'
S, SS	SS	SS'	SS	SS'
S', SS'	SS'	SS'	SS'	SS'

除了具有两种大小的数学符号, 数学模式内的 4 种字体尺寸的实际大小取决于数学模式外部当前的字体尺寸, 例如当前字体尺寸为 10 pt 时, D 与 T 尺寸也是 10 pt, S 尺寸是 7 pt, SS 尺寸是 5 pt.

§ 9.3 数学模式中的参数

在表格 9.2 中列出了数学模式中的一些参数, 它们的默认值能满足大多数用户的需要, 如果用户有特殊要求, 可以用命令 `\setlength` 重设它们的值. 表中提到的“长”的或“短”的行间公式不是公式本身的绝对长度, 而是与上方文本行的相对比较. 若上方文本行的结尾到页面左边界的距离大于公式左边界到页面左边界的距离, 则称公式是长公式, 否则称为短公式. 直观看, 被上方最末一行文本遮住一些公式就是长公式, 一点都未被遮住的就是短公式.

表中第 3 列是当基本字体尺寸为 10pt 时, 参数的默认值.

表 9.2 数学模式参数

参数	意义	默认值
<code>\arraycolsep</code>	array 环境中列间距宽度的一半	5.0pt
<code>\jot</code>	在 eqnarray 和 eqnarray* 环境增大或减小行间隔	3.0pt
<code>\abovedisplayskip</code>	在长行间公式与上方文本之间插入的竖直距离	8.5pt plus 3.0pt minus 4.0pt
<code>\belowdisplayskip</code>	在长行间公式与下方文本之间插入的竖直距离	8.5pt plus 3.0pt minus 4.0pt
<code>\abovedisplayshortskip</code>	在短行间公式与上方文本之间插入的竖直距离	0.0pt plus 2.0pt
<code>\belowdisplayshortskip</code>	在短行间公式与下方文本之间插入的竖直距离	4.0pt plus 2.0pt minus 2.0pt
<code>\mathindent</code>	选用了文档列选项 fleqn 时行间公式的缩进量	25.0pt

§ 9.4 定理定义的排版

在数学论文或书籍中常常需要用特定的格式展示定理、公理、命题、引理、定义等, 而且带有顺序编号, 为了做到这一点, 需要应用以下命令:

```
\newtheorem{定理环境名}{标题}[主计数器名]
```

例如

```
\newtheorem{theorem}{Theorem}[chapter]
```

定义一个名为 theorem 的新的定理环境, 每当这个环境被调用时, 先用黑体字打印 Theorem, 再打印一个自动生成的序号, 然后用斜体打印环境内部的文本. 方括号内的可选项应该是章节计数器的名字, 例如这里选的是 chapter, 指示把 chapter 计数器作为主计数器, 而定理计数器则作为从计数器, 每当章号发生改变时, 就要重置定理计数器的值. 当 theorem 环境被定义后, 就可用如下语句进入这个环境:

```
\begin{定理环境名}[附加标题] 文本 \end{定理环境名}
```

例如在定义了定理环境 theorem 后, 输入以下语句 (见 9-4-1.tex):

```
\begin{theorem}[Fermat] There do not exist integers  $n > 2$ ,  
 $x$ ,  $y$ , and  $z$  such that  $x^n + y^n = z^n$ . \end{theorem}
```

就能得到以下输出

Theorem 9.1 (Fermat) *There do not exist integers $n > 2$, x , y , and z such that $x^n + y^n = z^n$.*

当定理标题与附加标题含有中文时, 应使用黑体 \HEI, 并用花括号括起来; 当定理的正文含有中文时, 可指定使用直立楷体 \upshape\KAI, 并用花括号把这两条命令及中文括起来.

又如, 输入

```
\newtheorem{dingli}{\HEI定理}[chapter]  
\begin{dingli}[{\HEI费马}]{\upshape\KAI 不存在使得%  
~ $x^n + y^n = z^n$ ~的整数~ $n > 2$ 、~ $x$ 、~ $y$ ~以及~ $z$ .}  
\end{dingli}
```

其中字体命令 \HEI 的定义为 \newcommand*{\HEI}{\CJKfamily{hei}}, \KAI 的定义为 \newcommand*{\KAI}{\CJKfamily{kai}}. 其输出为

定理 9.1 (费马) 不存在使得 $x^n + y^n = z^n$ 的整数 $n > 2$ 、 x 、 y 以及 z .

`\newtheorem`的另一种用法是:

```
\newtheorem{定理环境名}[编号]{标题}
```

其中的选项 `编号` 是另一个定理环境的名字, 它与当前的环境共用同一个序号计数器. 例如:

```
\newtheorem{proposition}[theorem]{Proposition}
```

这个命令中间的选项是 `theorem`, 表示 `proposition` 环境与 `theorem` 环境合用一个定理计数器. 这样接在 **Theorem 9.1** 后面的是 **Proposition 9.2**, 而不是 **9.1**.

\LaTeX 有一个加强定理环境功能的宏包 `amsthm`, 只要在导言部分加入命令

```
\usepackage{amsthm}
```

就可以调用这个宏包. `amsthm` 宏包增加了以下功能:

1. 添加了星号命令 `\newtheorem*`, 以创立不带序号的定理环境;
2. 有 3 种预定义的定理格式可供选用:
`plain`: 标题与编号均为黑体, 定理正文用斜体;
`definition`: 标题与编号均为黑体, 定理正文用正常字体;
`remark`: 标题与编号用斜体, 正文用正常字体.
 选用格式的指令是 `\theoremstyle{格式}`, 此命令以后的 `\newtheorem` 命令建立的定理环境都采用这个格式, 直至遇到新的 `\theoremstyle` 命令为止.
3. 用户可以使用 `\newtheoremstyle` 命令建立自己的定理环境格式. 其用法可参见 `thmtest.tex` 或 `amsthm.dtx`.
4. 命令 `\swapnumbers` 可使其后创立的定理环境中的序号打印在标题之前, 例如: **1 Theorem**.
5. 对于较短的证明, 可以使用 `proof` 环境:

```
\begin{proof}[标题] ... \end{proof}
```

如果有可选项 [标题], 则它先印出这个标题(斜体), 如果没有这个可选项, 则它先印出默认标题 *Proof*. 在结束环境时自动生成证毕记号 \square . 如果不使用 `proof` 环境, 而是使用命令 `\proof`, 则也印出默认标题 *Proof*, 但需在证明的最后显式使用命令 `\qed` 打印出证毕记号. 用户可以用命令 `\qedsymbol` 自定义证毕记号.

对于中文证明, 可如下重定义证毕记号并使用中文标题“证.”:

```
\renewcommand*{\qedsymbol}{[证毕]}
\begin{proof}[{\HEI 证:}] ... \end{proof}
```

§ 9.5 巧妙使用阵列环境

利用 `array` 环境, 可以排版一些比较复杂的公式, 例如 (见 9-5-1.tex)

$$a \neq 0 \left\{ \begin{array}{l} b = 0 \left\{ \begin{array}{ll} c = 0, & y = f(x), \\ c \neq 0, & y = g(x), \end{array} \right. \\ b \neq 0, & y = h(x). \end{array} \right.$$

可以如下输入

```
\[
a\neq\ \bigg\{\begin{array}{l}b=0\ \Big\{
\begin{array}{ll}c=0, & y=f(x),\\
c\neq 0, & y=g(x),\end{array} \quad b\neq 0,\quad y=h(x).
\end{array}
\]
```

常见的三角形矩阵

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22} & \cdots & a_{2n} \\ & & \ddots & \vdots \\ 0 & & & a_{nn} \end{pmatrix}$$

可以如下输入

```
\[
\left(\begin{array}{cccc}
a_{11} & a_{12} & \cdots & a_{1n} \\
& a_{22} & \cdots & a_{2n} \\
& & \ddots & \vdots \\
0 & & & a_{nn}
\end{array}\right)
```

下面的矩阵可以看成是分块矩阵, 同时也是加边矩阵:

$$\left(\begin{array}{ccc|ccc} a & \cdots & a & b & \cdots & b \\ & \ddots & \vdots & & \ddots & \\ & & a & b & & \\ \hline & & 0 & c & \cdots & c \\ & & & \vdots & & \vdots \\ & & & c & \cdots & c \end{array} \right) \left. \begin{array}{l} \\ \\ \\ \end{array} \right\} p$$

$$\left. \begin{array}{l} \\ \\ \end{array} \right\} q$$

$$\underbrace{\hspace{1cm}}_m \quad \underbrace{\hspace{1cm}}_n$$

输入时把它看成是一个 2×2 矩阵 $\begin{smallmatrix} X & Y \\ Z & W \end{smallmatrix}$, 其中 X 是一个 6×6 矩阵, 即上面显示的分块矩阵; Y 是一个单列矩阵, 放右面的两个竖直角括号, 中间空一行; Z 是一个 1×2 矩阵, 放下面的两个水平花括号; W 是空白元素. 从整体看是矩阵嵌套. 输入如下, 其中 \@ -表达式 “ $\text{\@}\{\text{\hspace}\{-5\text{pt}\}}\}$ ” 的含义是: 去掉它所在的两个列之间原有的间隔, 代之以花括号内给出的命令, 这里就是 $\text{\hspace}\{-5\text{pt}\}$. 目的是为了拉近两列间的距离. 命令 $\text{\@}\{-5\text{pt}\}$ 是用于拉近两行之间的距离. 这里还出现了一个自定义的命令 \adots 表示上升点列, 是仿照 \ddots 命令定义的. 具体定义方法可参看 9-5-1.tex.

```
\[
\begin{array}{c@{\hspace{-5pt}}l}
\left(\begin{array}{ccc|ccc}
a & \cdots & a & b & \cdots & b \\
& \ddots & \vdots & & \ddots & \\
& & a & b & & \\
\hline
& & 0 & c & \cdots & c \\
& & & \vdots & & \vdots \\
& & & c & \cdots & c
\end{array}\right)
& \begin{array}{l} \\ \\ \end{array}
\end{array}
\left. \begin{array}{l} \\ \\ \end{array} \right\} p \\
& \left. \begin{array}{l} \\ \\ \end{array} \right\} q \\
\end{array}
\@{-5pt}
\begin{array}{cc}
\underbrace{\hspace{1cm}}_m & \underbrace{\hspace{1cm}}_n
\end{array}
&
\end{array}
```

array 环境不会自动产生公式编号, 若想显示公式编号, 需要使用 $\text{\$}\dots\text{\$}$ 作行间公式的标记, 并在结束数学模式的标记之前插入下列命令

\leqno 编号 或 \leqno 编号

其中 `\eqno` 将编号显示在公式右边, 对齐在页面右边界. `\leqno` 将编号显示在公式左边, 对齐在页面左边界. 例如输入

```


$$\left(\begin{array}{lcr} lll & ccc & rrr \\ l & c & r \end{array}\right) \eqno(5.3)$$


```

输出为

$$\left(\begin{array}{lcr} lll & ccc & rrr \\ l & c & r \end{array}\right) \quad (5.3)$$

这是给整个公式一个编号, 并且是人工编号, 若要自动编号, 或每行单独编号, 需使用 `equation` 或 `eqnarray` 环境.

数学公式中出现的多行下标可以看成是一个小型矩阵, 故可使用 `array` 环境排版多行下标. 通常的下标使用角标字体, 对于多行下标最好使用二级角标字体. 因下标命令对 `array` 环境不起作用, 所以需要显式指定字体尺寸, 并相应缩小行距. 这有两种做法, 一是在 `array` 环境中对每个元素指定字体尺寸, 并在换行符后加负的距离; 二是使用文字模式的字体尺寸, 既改变了字的大小, 又自动改变了行距. 第一种方法产生的下标过于靠下, 所以使用第二种方法. 但文字模式的字体尺寸声明不能放在数学模式内部, 因此要用 `\mbox` 进入文字模式, 但 `array` 只能出现在数学模式中, 所以还要用数学模式标记使 `array` 环境成为 `\mbox` 盒子中的行内公式. 下面给出一个例子, 输入

```


$$A = \sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n \\ 1 \leq k \leq p}} a_{ijk}$$


```

输出为

$$A = \sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n \\ 1 \leq k \leq p}} a_{ijk}$$

如果我们使用 `amsmath` 宏包的话, 对于多重角标有一个很简单的解决方法, 读者可参看第 151 页.

§ 9.6 多行公式左列问题

`eqnarray` 环境总是按三列排版公式, 如果左列的公式很长, 那么右列就很窄了. 一个变通的方法是在左列公式里选一个符号作为中间列, 把它后面的部分作为右列. 不过这样处理时, 原来左列公式的内部会出现较宽的列间距. 最好是有类似于表格环境中的 `\multicolumn` 命令, 把很长的左列单独放在一行上, 而又不影响下面各列的宽度. 遗憾的是在 `eqnarray` 环境中不能使用这个“多列合一”的命令. 好在 \LaTeX 提供了下述命令

```
\lefteqn{公式}
```

它显示作为参数的公式, 但被认为所占的宽度为零, 所在行左边列的列间距仍然存在. 通常在第一行使用这个命令. 例如输入 (见 9-6-1.tex)

```
\begin{eqnarray*}
\lefteqn{w+x+y+z = } \\\
& \& a+b+c+d+e+f+g+ \\\
& \& h+i+j+k+l+m+n \\
\end{eqnarray*}
```

输出为

$$\begin{array}{l} w+x+y+z = \\ \quad a+b+c+d+e+f+g+ \\ \quad h+i+j+k+l+m+n \end{array}$$

后两行缩进的距离实际就是列间距. 通过在 `\lefteqn{...}` 和 `\\` 之间插入 `\hspace*{长度}` 可以调整缩进深度. 正的长度增加缩进, 负值则减少缩进.

§ 9.7 amsmath 宏包简介

本章介绍的 `amsmath` 宏包为 2.13 版本, 日期为 2000/07/18. 不同的版本稍有区别, 例如 1.2b 版本没有 `Bmatrix` 环境, `equation` 环境可以排版出多行公式, 而 2.13 版本在这两个环境上正好相反. 当用户不能正常使用本章介绍的命令时, 应检查所用版本是否太旧.

`amsmath` 宏包会自动调入 `amsopn`、`amstext` 和 `amsbsy` 等几个宏包 (但它不会自动调入 `amsfonts` 宏包), 如果仅仅需要这几个宏包提供的为数不多的功能, 也可

不通过 `amsmath` 而单独调入它们. 为叙述简单, 下面提到 `amsmath` 的功能时一般不再说明是其本身的还是它调入的其他宏包的功能.

`amsmath` 及其调入的宏包提供了许多数学符号和函数名, 还提供如下一些环境用于排版行间公式:

<code>equation</code>	<code>equation*</code>	<code>align</code>	<code>align*</code>
<code>gather</code>	<code>gather*</code>	<code>flalign</code>	<code>flalign*</code>
<code>multline</code>	<code>multline*</code>	<code>alignat</code>	<code>alignat*</code>
<code>split</code>			

除了 `split`, 其他环境都有两种形式, 不带星号的环境具有自动编号的功能, 带有星号时不参与自动编号. 在多行公式的换行命令 `\\` 之前写上 `\notag` 可使该行不带编号. 在两种环境中都可以使用 `\tag{标号}` 人为指定公式的标号, 该标号自动被圆括号括起来, 若使用 `\tag*{标号}`, 则标号不会自动带圆括号. 此处所谓的标号可以是任何字母、数字或其他符号.

除了 `split` 环境之外, 其他环境都进入行间公式模式. `split` 环境不能单独使用, 必须放在其他几个数学环境内部, 但不可用在 `multline` 环境内.

L^AT_EX 的 `eqnarray` 环境仍然可用, 但最好是使用 `align` 或者 `equation` 与 `split` 代替之, 它们不像 `eqnarray` 在对齐位置产生过大的间隔.

为了调入 (加载) 这个 `amsmath` 宏包, 须在导言区写上语句

`\usepackage[选项]{amsmath}`

通常可不写选项 (即实际使用默认选项). 选项的名称和含义简介如下, 其中具有相反意义的选项放在一起介绍, 默认选项放在前面, 显然这类成对的选项不能同时选用.

centertags, tbtags 对于 `split` 环境 (见第 143 页). 公式编号默认排在上下居中的位置. 若使用 `tbtags` 选项, 则当公式编号位于左侧时, 编号将被排在公式首行的左侧, 当公式编号位于右侧时, 编号将被排在公式末行的右侧.

sumlimits, nosumlimits 在行间公式中, 求和号的上下限默认排在 \sum 符号的上方和下方. 选项 `nosumlimits` 使上下限排在 \sum 符号右侧. 在具有两种尺寸的符号中, 除了积分号之外, 其他符号均受到该选项的影响.

nointlimits, intlimits 积分号 \int 的上下限默认排在右侧. `intlimits` 选项使上下限排在各种积分符号的上方和下方.

namelimits, nonamelimits 在行间公式中, 对于函数名 `\det`, `\gcd`, `\inf`, `\lim`, `\liminf`, `\limsup`, `\max`, `\min`, `\Pr` 以及 `\sup`, 默认极限过程符号排在这些函数名的下方, 形如 $\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$. 若使用选项 `nonamelimits`, 则极限过程排在上述函数名的右下角, 形如 $\lim_{n \rightarrow \infty} x_n$.

reqno, **leqno** 在行间公式中, 公式编号默认排在公式右侧, 对齐在页面右边界处. 选项 **leqno** 使公式编号排在公式左侧.

fleqn 该选项使所有居中对齐的行间公式左对齐, **\mathindent** 的值决定了公式左端与页面左边界之间的距离. 若不使用该选项, 则公式默认是居中对齐的.

上述第 2、3、4 对选项决定了行间公式上下限的标准位置, 可以通过在公式中使用命令 **\limits** 或 **\nolimits** 屏蔽掉 选项在该公式中的作用, 而使上下限出现在函数名的上下方或右侧. 对于行内公式, 上下限通常排在符号右侧, 同样可以使用上述两个命令人为改变上下限的位置. 例如

$$\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$$

总是排成 $\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$, 而

$$\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$$

总是排成 $\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$.

如果在文档类别的选项中使用了选项 **leqno**、**fleqn**, 则不必在宏包 **amsmath** 的选项中重复.

若只使用宏包 **amsmath**, 会发现 **\boldsymbol** 命令对具有两种尺寸的符号不起作用, 此时对这些符号使用 **\pmb** 命令, 就可将它们排成粗体. **pmb** 意即 poor man's bold, 它并不使用粗体字库, 而是通过微小平移多次重复打印一个符号产生粗体效果, 打印质量略显逊色.

下文中在节标题后附有 (**amsmath**) 时, 介绍的新命令均为加载 **amsmath** 宏包后可以使用的命令. 注意, \TeX 中的分式型命令 **\atop** 和 **\choose** 不能再用, 可重新定义 (见第 156 页) 或使用新的命令如 **\binom** (见第 155 页).

§ 9.8 公式中的文本 (**amsmath**)

在公式中插入文本可以使用命令

$$\text{\text{文本}}$$

该命令比 \LaTeX 原有命令 **\mbox** 功能更强一些, 它会按照所处位置自动改变字体大小. 例如为了排板输出成 $\text{This is}_{\text{subscript}}$, 使用 **\text** 与 **\mbox** 应分别如下输入

$$\text{This is}_{\text{\text{subscript}}}$$

$$\text{This is}_{\text{\mbox{\scriptsize subscript}}}$$

在多行公式的两行之间插入文本可以使用命令

$$\text{\intertext{文本}}$$

该命令上方和下方的公式仍处于同一个数学环境之中, 因此可以保持上下的对齐关系不变. 例如输入 (见 9-8-1.tex)

```

\begin{align*}
(x+\mathrm{i}y)(x-\mathrm{i}y) &= x^2 + \mathrm{i}xy - \mathrm{i}xy - \mathrm{i}^2y^2 \\
&= x^2 + y^2 \\
\intertext{利用~$\mathrm{i}^2=-1$, 还可得到}
(x+\mathrm{i}y)^2 &= x^2 + 2\mathrm{i}xy - y^2 \\
(x-\mathrm{i}y)^2 &= x^2 - 2\mathrm{i}xy - y^2
\end{align*}

```

输出为

$$\begin{aligned}
 (x + \mathrm{i}y)(x - \mathrm{i}y) &= x^2 + \mathrm{i}xy - \mathrm{i}xy - \mathrm{i}^2y^2 \\
 &= x^2 + y^2
 \end{aligned}$$

利用 $\mathrm{i}^2 = -1$, 还可得到

$$\begin{aligned}
 (x + \mathrm{i}y)^2 &= x^2 + 2\mathrm{i}xy - y^2 \\
 (x - \mathrm{i}y)^2 &= x^2 - 2\mathrm{i}xy - y^2
 \end{aligned}$$

§ 9.9 单个公式 (amsmath)

单个公式是指整个公式是一个整体, 最多只有一个自动公式编号(使用不带星号的环境), 或没有自动编号(使用带有星号的环境). 单个公式可以只有一行, 也可以有多行. 当单个公式只有一行时, 自动编号时使用 `equation` 环境, 不参与自动编号时使用 `equation*` 环境, 这相当于 L^AT_EX 原有的 `displaymath` 环境.

当单个公式很长, 或不适合排在一行时, 可以把单个公式排成多行, 有如下两种方法.

第一种方法是在 `equation` 环境中使用 `split` 环境, 使每行公式在指定位置对齐, 见下例, 对齐位置用 `&` 指定, 换行符用 `\\`. 输入(见 9-9-1.tex)

```

\begin{equation}
\begin{split}
(a+b)^2 &= a^2 + b^2 + 2ab \\
(a+b+c)^2 &= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc
\end{split}
\end{equation}

```

输出为

$$\begin{aligned}(a+b)^2 &= a^2 + b^2 + 2ab \\ (a+b+c)^2 &= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc\end{aligned}\tag{9.1}$$

如果不写对齐符号, 则输出为右对齐

$$\begin{aligned}(a+b)^2 &= a^2 + b^2 \\ (a+b+c)^2 &= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc\end{aligned}\tag{9.2}$$

使用 `split` 环境后, 公式编号位于上下居中的位置.

第二种方法是使用 `multline` 环境, 它使第一行靠左, 最末行靠右, 其他行居中对齐. 首末两行与左右边界的距离由 `\multlinegap` 的值决定, 可用 `\setlength` 或 `\addtolength` 改变其值. 公式编号总是位于首行左端或末行右端. 使用命令 `\shoveleft` 或 `\shoveright` 可使中间的公式靠左或靠右对齐. 例如输入

```
\begin{multline}
\framebox[.5\columnwidth]{第一行(自动靠左)}\\
\framebox[.4\columnwidth]{自动居中}\\
\boxed{\sum_{k=1}^{100} k = 5050}\\
\shoveleft{\framebox[.7\columnwidth]{强制靠左}}\\
\shoveright{\framebox[.7\columnwidth]{强制靠右}}\\
\framebox[.5\columnwidth]{第末行(自动靠右)}
\end{multline}
```

输出为

$$\begin{array}{c} \text{第一行(自动靠左)} \\ \text{自动居中} \\ \boxed{\sum_{k=1}^{100} k = 5050} \\ \text{强制靠左} \\ \text{强制靠右} \\ \text{第末行(自动靠右)} \end{array}\tag{9.3}$$

从该例可见, 给公式加框实际是用命令

$\boxed{\text{公式}}$

如果使用了选项 `fleqn`, 则居中对齐的行都会变为左对齐.

§ 9.10 方程组 (amsmath)

方程组由多个公式组成, 其中每个公式可以占一行也可能占多行. 区分单个公式和方程组, 不能只看行数, 因为单个公式也可以占有多行. 如果整个公式只能有一个自动编号, 则为单个公式, 如果能有多个自动编号, 则为方程组. 两者所用环境不同, 但都有相应不参与自动编号的环境 (带星号的环境).

9.10.1 gather 环境

在 `gather` 环境中每行公式用 `\\` 分隔, 不指定上下对齐的位置. 通常各个公式都是居中对齐, 但若在加载 `amsmath` 时使用了选项 `fleqn`, 则各个公式都是左对齐. 下例中有三个公式, 为了便于看清结构, 公式内容都是文字. 其中一个是 `split` 环境, 该环境是带对齐标记的. 输入 (见 9-10-1.tex)

```
\begin{gather}
  \text{这是第一个公式}\\
  \begin{split}
    \text{这是第二个公式, } & \text{使用~split~环境}\\
    & \text{占据了两行}
  \end{split}\\
  \text{这是第三个公式}
\end{gather}
```

输出为

这是第一个公式	(9.4)
这是第二个公式, 使用 <code>split</code> 环境 占据了两行	(9.5)
这是第三个公式	(9.6)

在 `gather` 环境中, 使用命令 `\notag` 可使某些行不编号, 例如

```
\begin{gather}
  (a+b)^2=a^2+b^2 \notag \\
  (a+b+c)^2=a^2+b^2+c^2+2ab+2ac+2bc
\end{gather}
```

输出为

$$\begin{aligned}(a+b)^2 &= a^2 + b^2 \\ (a+b+c)^2 &= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc\end{aligned}\quad (9.7)$$

9.10.2 align 环境

利用 align 环境可以指定各行公式上下对齐的位置, 通常多在等号或关系运算符处对齐. 例如输入 (见 9-10-2.tex)

```
\begin{align}
&\text{圆周率}\pi \approx 3.1415926535897932384626\\
&\text{自然对数的底}e \approx 2.718281828459045
\end{align}
```

输出为

$$\begin{aligned}\text{圆周率}\pi &\approx 3.1415926535897932384626 \\ \text{自然对数的底}e &\approx 2.718281828459045\end{aligned}\quad \begin{aligned}(9.8) \\ (9.9)\end{aligned}$$

align 环境也可使几组公式并排在一起, 此时在同一行上出现分别属于不同组的几个公式, 每个组的公式内应有一个对齐符号 &, 用于该组的上下对齐, 同时在不同组的公式之间也要插入 & 符号, 以分隔这些公式. 同一行上若有 n 个公式, 就必须有 $n+(n-1) = 2n-1$ 个 &, 把一行分成 $2n$ 列. 奇数列总是靠右对齐, 偶数列总是靠左对齐, 于是同一组的公式就靠在一起了. 有时为了叙述方便, 就将这种左右靠在一起的两列称为是一个“列对”. 输入

```
\begin{align*}
(x^n)' &= nx^{n-1} & (\sin x)' &= \cos x \\
(a^x)' &= a^x \ln a & (\cos x)' &= -\sin x \\
&& (\tan x)' &= \frac{1}{\cos^2 x}
\end{align*}
```

输出为

$$\begin{array}{ll}
 (x^n)' = nx^{n-1} & (\sin x)' = \cos x \\
 (a^x)' = a^x \ln a & (\cos x)' = -\sin x \\
 & (\tan x)' = \frac{1}{\cos^2 x}
 \end{array}$$

9.10.3 flalign 环境

`flalign` 环境与 `align` 环境的语法完全一样, 仅仅是输出效果有区别: `flalign` 环境在同行多个公式之间插入足够的空白以充满一行, 精确地说就是在一行上对应于偶数个 `&` 的位置自动插入弹性长度, 以充满一行. 例如将上例的环境改为 `flalign*`, 则输出为

$$\begin{array}{ll}
 (x^n)' = nx^{n-1} & (\sin x)' = \cos x \\
 (a^x)' = a^x \ln a & (\cos x)' = -\sin x \\
 & (\tan x)' = \frac{1}{\cos^2 x}
 \end{array}$$

9.10.4 alignat 环境

在 `align` 环境中, 同一行几个列对之间的间距使用默认值 (类似于阵列环境的列间隔), 在 `flalign` 环境中, 这个间距是个弹簧, 总是尽量大, 而在 `alignat` 环境中, 这个间距默认是零, 因此可以通过插入空白长度使列对之间保持指定的间隔. `alignat` 环境需要一个参数, 其值表示同一行中列对的个数, 从前面的介绍可知, 这个值等于同一行中 `&` 的个数加 1 后再除以 2. 例如将上面例子的环境改为 `\begin{alignat*}{2}...\end{alignat*}`, 并且插入间隔, 即如下输入 (见 9-10-3.tex, 其中的 `\label{eq:x}` 是为别处引用所设, 与本环境无关)

```

\begin{alignat}{2}
(x^n)' & \&= nx^{n-1} & \&\hspace{20pt} \\
(\sin x)' & \&= \cos x \\
(a^x)' & \&= a^x \ln a & \& (\cos x)' \&= -\sin x \\
& \& \& (\tan x)' \&= \frac{1}{\cos^2 x} \label{eq:x} \\
\end{alignat}

```

输出为

$$(x^n)' = nx^{n-1} \quad (\sin x)' = \cos x \quad (9.10)$$

$$(a^x)' = a^x \ln a \quad (\cos x)' = -\sin x \quad (9.11)$$

$$(\tan x)' = \frac{1}{\cos^2 x} \quad (9.12)$$

有时需要在公式后面附加一些上下对齐的文字, 可如下输入

```
\begin{alignat}{2}
y &= f(x)+g(x) &\&\quad &\&\text{(由引理\,2)}\\
&= \sec^2 x &\&\quad &\&\text{(由\eqref{eq:x}式)}
\end{alignat}
```

输出为

$$y = f(x) + g(x) \quad (\text{由引理 2}) \quad (9.13)$$

$$= \sec^2 x \quad (\text{由(9.12)式}) \quad (9.14)$$

上述两个例子也示例了方程编号的引用方法, 在被引用的方程处写上标记 `\label{标记}`, 在引用处写上 `\eqref{标记}`, 此处标记可以是任何字符串. 将文件编译两次后, 引用处的 `\eqref{标记}` 就会输出为相应方程的编号. 引用命令 `\eqref` 也可改用 `\ref`, 两者的区别是输出时前者自动被圆括号括起来, 而后者并不会自动添加括号. 若引用时用 `\pageref{标记}`, 输出后是被引用标记所在页的页码.

9.10.5 gathered, aligned 和 alignat 环境

前面介绍的方程组环境 `gather`、`align` 和 `alignat`, 无论公式的实际宽度是多大, 它们产生的结构总是占满一行的宽度, 因此不能在它们周围添加括号, 而很多方程组是用花括号连成一组的, 为解决这类问题, 可以使用以 `ed` 结尾的环境 `gathered`、`aligned` 和 `alignated`, 它们的语法和效果虽和不以 `ed` 结尾的环境相同, 但整个结构只占有公式本身实际的宽度, 而不是占满整行, 这样它们就可作为一个“块”结构而放在其他环境之中, 整体作为一个特大的符号与其他符号一同处理. 这些结构还可加上位置参数, 以决定与其他符号的竖直对齐方式. 需要注意的是这些以 `ed` 结尾的环境本身不再有自动编号的功能, 此外在旧版本中 `aligned` 环境中每行只能有一个对齐符号 `&`, 并且没有 `alignated` 环境. 下面是一个例子 (见 9-10-4.tex):


```

\begin{equation}
\begin{aligned} a&=b+c \\ d&=bb+cc \end{aligned}
\Longrightarrow
\begin{gathered} [b] \ A=aa+bb \\ D=c+f \end{gathered}
\Longrightarrow
\begin{aligned} [t] \ X&=A+aa \\ Y&=D+d \end{aligned}
\end{equation}

```

输出为

$$\begin{array}{rcl}
 a = b + c & & A = aa + bb \\
 d = bb + cc & \Rightarrow & D = c + f \Rightarrow X = A + aa \\
 & & Y = D + d
 \end{array} \quad (9.15)$$

可以在上述块结构旁加各种定界符, 例如输入

```

\begin{equation*}\left.\begin{aligned}
B'&=-\partial\mathrm{times} E \\
E'&=-\partial\mathrm{times} B-4\pi j
\end{aligned}\right\} \quad \text{Maxwell 方程}
\end{equation*}

```

输出为

$$\left. \begin{array}{l} B' = -\partial \times E \\ E' = \partial \times B - 4\pi j \end{array} \right\} \quad \text{Maxwell 方程}$$

9.10.6 cases 环境

cases 环境专门用于排版左侧带有花括号的方程组, 仅举一例:

```

\begin{equation} f(x)=
\begin{cases} 1 & -1 < x < 1 \\ 0 & \text{其他} \end{cases}
\end{equation}

```

输出为

$$f(x) = \begin{cases} 1 & -1 < x < 1 \\ 0 & \text{其他} \end{cases} \quad (9.16)$$

§ 9.11 矩阵 (amsmath)

`matrix` 环境用于排版矩阵, 本身不带定界符. 如果需要带定界符的矩阵, 应使用 `pmatrix`、`bmatrix`、`Bmatrix`、`vmatrix` 和 `Vmatrix` 等环境, 它们分别带有定界符 `()`、`[]`、`{}`、`||` 和 `|||`. 默认矩阵最多有 10 列, 各列均居中对齐. 与 L^AT_EX 原有的 `array` 阵列环境 (见第 56 页) 一样, 这些环境必须放在数学模式之中.

可以使用命令

```
\setcounter{MaxMatrixCols}{数}
```

或

```
\addtocounter{MaxMatrixCols}{数}
```

改变最大列数的默认值. 如果想要改变列的对齐方式, 必须使用 `array` 阵列环境. 在低版本 `amsmath` 宏包中没有定义 `Bmatrix` 环境, 用户可自行如下定义:

```
\newenvironment{Bmatrix}%
{\left\lbrace\matrix}\endmatrix\right\rbrace
```

下面例子整体是一个无边的矩阵, 每个元素也都是一种矩阵 (见 9-11-1.tex):

```
\[ \begin{matrix}
\begin{matrix} 1 & 2 \end{matrix} & \begin{matrix} 3 & 4 \end{matrix} \\
\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} & \begin{pmatrix} 3 & 4 \\ 1 & 2 \end{pmatrix} \\
\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} & \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix} \\
\begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix} & \begin{Bmatrix} 3 & 4 \\ 1 & 2 \end{Bmatrix} \\
\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} & \begin{vmatrix} 3 & 4 \\ 1 & 2 \end{vmatrix} \\
\begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix} & \begin{Vmatrix} 3 & 4 \\ 1 & 2 \end{Vmatrix}
\end{matrix} \]
```

输出为

$$\begin{matrix} \begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} & \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} & \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \\ \begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix} & \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} & \begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix} \end{matrix}$$

从上述例子可见, 可用

```
\\[距离]
```

调整相邻行之间的距离, 对于其他行间公式, 也可如此改变公式的行间隔.

对于出现在行内的矩阵, 一般希望它小一些, 以免撑大行宽, 此时可考虑使用 `smallmatrix` 环境. 例如行内矩阵 $\begin{pmatrix} a & b & c \\ x & y & z \end{pmatrix}$ 是如下输入的:

```
\bigl( \begin{smallmatrix} a & b & c \\ x & y & z \end{smallmatrix} \bigr)
```

§9.12 多重数学符号 (amsmath)

9.12.1 多重角标

1 多重角标命令

命令 `\substack` 可以排版多重上标或下标, 两行角标之间用 `\\` 分隔, 排版后角标是居中对齐的. 例如输入 (见 9-12-1.tex)

```
\begin{equation}
\sum_{\substack{k_0, k_1, \ldots > 0 \\ k_0 + k_1 + \cdots = n}} F(k_i)
\end{equation}
```

输出为

$$\sum_{\substack{k_0, k_1, \dots > 0 \\ k_0 + k_1 + \dots = n}} F(k_i) \quad (9.17)$$

2 多重角标环境

多重角标环境 `subarray` 不但可以排版多重上标或下标, 而且可以指定角标左对齐 (使用参数 `l`) 或居中对齐 (使用参数 `c`). 下例的角标与上一例相同, 但指定是左对齐. 输入

```
\begin{equation*}
\sum_{\begin{subarray}{l} k_0, k_1, \ldots > 0 \\ k_0 + k_1 + \cdots = n \end{subarray}} F(k_i)
\end{equation*}
```

输出为

$$\sum_{\begin{subarray}{l} k_0, k_1, \dots > 0 \\ k_0 + k_1 + \dots = n \end{subarray}} F(k_i)$$

9.12.2 多重积分

多重积分符号 `\iint`、`\iiint`、`\iiiiiint` 和 `\idotsint` 与普通积分符号 `\int` 一样具有两种尺寸, 上下限既可放在右侧也可放在上下方. 下面是几个例子. 输入

```
\begin{gather*}
\iint\limits_D f(x,y)\,,\dif x\dif y\qqquad
\iiint\limits_V f(x,y,z)\,,\dif V \\\
\iiiiiint\limits_G f(x,y,z,t)\,,\dif G\qqquad
\idotsint\limits_U f(x_1,\dots,x_n)\,,\dif U
\end{gather*}
```

输出为

$$\iint_D f(x,y) \, dx dy \quad \iiint_V f(x,y,z) \, dV$$

$$\iiiiiint_G f(x,y,z,t) \, dG \quad \int \cdots \int_U f(x_1, \dots, x_n) \, dU$$

9.12.3 叠置重音符号

数学符号上的重音符号, 如 \hat{A} 或 \tilde{A} 可输入为 `\hat{A}` 和 `\tilde{A}`. 数学重音符号可以叠置, 例如输入 `\hat{\tilde{A}}` 可以产生双层重音符号. 当所用 `amsmath` 宏包版本较低时, 叠置的重音符号不会对齐. 为解决这种问题, 宏包中提供了一组用于叠置的重音命令, 与原有的数学重音命令同名, 但第一个字母大写, 它们是

<code>\Hat</code>	<code>\Breve</code>	<code>\Grave</code>	<code>\Bar</code>	<code>\Dot</code>
<code>\Check</code>	<code>\Acute</code>	<code>\Tilde</code>	<code>\Vec</code>	<code>\Ddot</code>

上面的例子若输入成 `\Hat{\Tilde{A}}`, 则输出是 $\hat{\tilde{A}}$. 更多层的叠置也是可以的.

原有的双点重音符号 (`\ddot`) 可用来表示对时间的二阶导数, `amsmath` 宏包又提供了可以表示三阶和四阶时间导数的重音符号:

`\ddd\dot` `\dddd\dot`

输入 `\ddd\dot{u}` 和 `\dddd\dot{u}` 输出为 \dddot{u} 和 $\dddd\dot{u}$.

9.12.4 省略号

1. 三个点的省略号

L^AT_EX 中的省略号命令 `\ldots` 和 `\cdots` 仍然可用, 它们分别产生位于基线上和在一行内上下居中的三个点. 调入 `amsmath` 后, 增加了下列几个省略号命令:

<code>\dots</code>	<code>\dotso</code>	<code>\dotsc</code>	<code>\dotsm</code>	<code>\dotsi</code>
--------------------	---------------------	---------------------	---------------------	---------------------

当 `\dots` 位于式子中间时, 它会根据尾随其后的符号自动决定省略号的位置: 若后面是关系运算符如等号或二元运算符如 $+$ 、 $-$, 它相当于 `\cdots`, 其他情况相当于 `\ldots`. 见下例:

$$\begin{array}{lll}
 \$x_1 + x_2 + \dots + x_n\$ & \Rightarrow & x_1 + x_2 + \cdots + x_n \\
 \$x_1, x_2, \dots, x_n\$ & \Rightarrow & x_1, x_2, \dots, x_n \\
 \$x_i, \dots, x_j - \dots, x_k\$ & \Rightarrow & x_i, \cdots + x_j - \cdots, x_k
 \end{array}$$

但要注意, 当 `\dots` 位于式子末尾时, 没有尾随符号决定它的位置, 它就把省略号排在了基线上, 此时应使用其他几个省略号命令以指定省略号的竖直位置, 命令的最后一个字母表明了命令的含义: `\dotso` 中的 `o` 表示 *ordinal* (二元的), 即相当于后面有一个二元运算符, 因此省略号是上下居中的; `\dotsc` 中的 `c` 表示 *comma* (逗号), 不是 *center*, 因此省略号是排在基线上的; `\dotsm` 中的 `m` 表示 *multiplication* (乘法), 因此相当于是 `\dotso`, 只不过它是用在乘式中, 省略号是上下居中的, 如 `$A_1A_2\dotsm$` 生成 $A_1A_2\cdots$; `\dotsi` 中的 `i` 表示 *integral* (积分), 省略号对齐在积分号的中部.

2. 长省略号

在矩阵中有时需要跨越几列的省略号, 在第 57 页使用过. 加载 `amsmath` 后可使用更简单的命令

<code>\hdotsfor{列数}</code>

例如输入

<pre> \[\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hdotsfor{4} \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}\] </pre>

可以得到

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

不过按照我国的出版标准, 规范形式应为

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

其输入方式是

```
\[ \begin{pmatrix}
a_{11} & a_{12} & \dots & a_{1n} \\
a_{21} & a_{22} & \dots & a_{2n} \\
\vdots & \vdots & & \vdots \\
a_{n1} & a_{n2} & \dots & a_{nn}
\end{pmatrix} \]
```

§ 9.13 分式 (amsmath)

9.13.1 普通分式

LaTeX 中的分式命令 `\frac` 仍然可用, 这个分式会随着所处环境而自动改变尺寸. 调入 `amsmath` 后, 增加了两个分式命令

```
\tfrac{分子}{分母}      \dfrac{分子}{分母}
```

它们分别表示在尺寸 `\textstyle` 和 `\displaystyle` 下打印分数. 例如, 无论在行内公式还是在行间公式, `\tfrac{a}{b}` 总是输出成 $\frac{a}{b}$, 而 `\dfrac{a}{b}` 总是输出成 $\frac{a}{b}$.

9.13.2 连分式

调入 `amsmath` 后, 增加了连分式命令

```
\cfrac[位置]{分子}{分母}
```

命令的含义是 **continued fraction**, 其中的可选项位置用于指定分子在分数线上的位置: 省略时是居中, l 表示与分数线左对齐, r 表示与分数线右对齐. 分母中仍可含有 `\cfrac` 命令, 分子分母自动使用同样大小的字体. 例如输入 (见 9-13-1.tex)

```
\[
  a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 +
    \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}
\]
```

输出为

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

9.13.3 二项式系数

调入 `amsmath` 后, 不能再使用 $\mathrm{T}_\mathrm{E}\mathrm{X}$ 中命令 `\atop` 和 `\choose`, 否则编译时会显示出类似下面的出错信息

```
! Package amsmath Error: Foreign command \atop;
  use \frac or \genfrac instead.
```

代替 `\choose` 的是命令:

```
\binom{分子}{分母}
\tbinom{分子}{分母}
\dbinom{分子}{分母}
```

它们都生成二项式系数表达式, 第一个命令会自动选择字体尺寸, 后两个则把当前激活尺寸分别当成是 `\textstyle` 和 `\displaystyle`. 举例如下:

$$\backslash \binom{n+1}{k} \Rightarrow \binom{n+1}{k}$$

9.13.4 自定义分式类命令

`amsmath` 提供了一个广义分式命令

```
\genfrac{左定界符}{右定界符}{线粗细}{字尺寸}{分子}{分母}
```

其中左定界符和右定界符是放在生成的分式左右两侧的定界符; 线粗细指定分数线的粗细, 空白时表示使用 L^AT_EX 中标准的分数线; 字尺寸指定所用字体的大小, 可为空白或数字 0-3, 空白时表示自动选择字体大小, 数字 0-3 分别表示当前激活字体尺寸为 `\displaystyle`、`\textstyle`、`\scriptstyle` 和 `\scriptscriptstyle`. 前面出现的分数类的命令是如下定义的 (实际定义更复杂一些):

```
\newcommand{\frac}{\genfrac{}{}{}{分子}{分母}}
\newcommand{\dfrac}{\genfrac{}{}{}0{分子}{分母}}
\newcommand{\tfrac}{\genfrac{}{}{}1{分子}{分母}}
\newcommand{\binom}{\genfrac(){}{}{分子}{分母}}
\newcommand{\dbinom}{\genfrac(){}{}0{分子}{分母}}
\newcommand{\tbinom}{\genfrac(){}{}1{分子}{分母}}
```

模仿这些例子, 可以如下重新定义 `\atop` 命令:

```
\renewcommand{\atop}[2]{\genfrac{}{}{0pt}{}{#1}{#2}}
```

注意这与原来的 `\atop` 的 T_EX 风格的用法不同了, 变成了 L^AT_EX 风格——分子与分母都是命令后面的参数, 不能放在命令的两边. 公式 (9.17) 现可如下输入

```
\begin{equation}
\sum_{\atop{k_0,k_1,\ldots>0}}{k_0+k_1+\cdots=n} F(k_i)
\end{equation}
```

§ 9.14 函数 (算子) 名 (amsmath)

在数学公式中, 通常将变量名排成斜体, 将函数名排成正体, 并在函数名的左右带有适当的空白.

9.14.1 已定义的函数名

已定义的函数名 (算子名) 有

<code>\arccos</code>	<code>arccos</code>	<code>\arcsin</code>	<code>arcsin</code>	<code>\arctan</code>	<code>arctan</code>
<code>\arg</code>	<code>arg</code>	<code>\cos</code>	<code>cosh</code>	<code>\cosh</code>	<code>cosh</code>
<code>\cot</code>	<code>cot</code>	<code>\coth</code>	<code>coth</code>	<code>\csc</code>	<code>csc</code>
<code>\deg</code>	<code>deg</code>	<code>\det</code>	<code>det</code>	<code>\dim</code>	<code>dim</code>
<code>\exp</code>	<code>exp</code>	<code>\gcd</code>	<code>gcd</code>	<code>\hom</code>	<code>hom</code>
<code>\inf</code>	<code>inf</code>	<code>\injl</code>	<code>injl</code>	<code>\ker</code>	<code>ker</code>
<code>\lg</code>	<code>lg</code>	<code>\lim</code>	<code>lim</code>	<code>\liminf</code>	<code>lim inf</code>
<code>\limsup</code>	<code>limsup</code>	<code>\ln</code>	<code>ln</code>	<code>\log</code>	<code>log</code>

<code>\max</code>	max	<code>\min</code>	min	<code>\Pr</code>	Pr
<code>\prolim</code>	prolim	<code>\sec</code>	sec	<code>\sin</code>	sin
<code>\sinh</code>	sinh	<code>\sup</code>	sup	<code>\tan</code>	tan
<code>\tanh</code>	tanh	<code>\varlimsup</code>	$\overline{\lim}$	<code>\varinjlim</code>	\varinjlim
<code>\varliminf</code>	\varliminf	<code>\varprojlim</code>	\varprojlim		

9.14.2 定义新的函数名

为了定义新的函数名, amsmath 提供了命令 (放在导言区)

```
\DeclareMathOperator{\函数名命令}{函数名}
\DeclareMathOperator*{\函数名命令}{函数名}
```

其中第二种 (带星号的) 形式定义的函数名类似于 `\lim`, 会根据所处环境或使用命令 `\limits` 将上下限放置在函数名的上方或下方. 函数名命令与函数名的文字可以不同. 例如在导言区写上命令

```
\DeclareMathOperator{\abc}{abc}
\DeclareMathOperator*{\uvw}{xyz}
```

则有

```
$\abc_1^2 \quad \quad \quad \abc\limits_1^2$      ⇒       $abc_1^2$     $\abc_1^2$ 
$\uvw_1^2 \quad \quad \quad \uvw\limits_1^2$      ⇒       $xyz_1^2$     $xyz_1^2$ 
```

对于某些临时用到的未定义过的函数名, 可直接在公式内部使用命令

```
\operatorname{函数名}           或           \operatorname*{函数名}
```

产生函数名, 星号的含义与前面定义中的星号含义相同. 例如

```
$\operatorname{abc}_a^b$      ⇒       $abc_a^b$ 
$\operatorname*{abc}\limits_a^b$      ⇒       $abc_a^b$ 
```

§ 9.15 其他功能 (amsmath)

9.15.1 公式中的空白间隔

在数学公式中产生空白间隔的一般命令是

```
\mspace{数mu}
```

其中单位mu是固定的,不能改用其他单位,也不可省略, $1\text{mu} = 1/18\text{em}$. 看一个例子,仅为示例`\mspace`的作用:

$$\sum \mspace{-13mu} a \mspace{9mu} b \ c \$ \Rightarrow \sum a \ bc$$

L^AT_EX 提供了几个数学空白命令, `amsmath` 又增加了几个空白命令,并定义了对应的长命令名,一同列表如下:

短命令	长命令	空白宽度大小
<code>\,</code>	<code>\thinspace</code>	U
<code>\!</code>	<code>\negthinspace</code>	U
<code>\:</code>	<code>\medspace</code>	U
	<code>\negmedspace</code>	U
<code>\;</code>	<code>\thickspace</code>	U
	<code>\negthickspace</code>	U
<code>_</code>		U
<code>\quad</code>		U
<code>\qquad</code>		U

9.15.2 调整根式指数的位置

`amsmath` 宏包提供了两个命令

`\leftroot{数}` `\uproot{数}`

用于调整根式指数的位置,移动的距离单位是一个很小的内部长度,但不可写上单位. `\leftroot` 用于指数的左右移动,正值为向左移动,负值为向右移动. `\uproot` 用于指数的上下移动,正值为向上移动,负值为向下移动. 示例如下 (见 9-15-1.tex):

$$\begin{aligned} \$\sqrt[\beta]{f(x)}\$ &\Rightarrow \sqrt[\beta]{f(x)} \\ \$\sqrt[\leftroot{-2}\uproot{2}\beta]{f(x)}\$ &\Rightarrow \sqrt[\beta]{f(x)} \end{aligned}$$

9.15.3 调整公式编号的竖直位置

当一行公式较长,使得在同一行上放不下公式编号时,公式编号将单独占一行被排在公式下方,此时可在该行公式末(换行符之前)用命令

`\raisetag{高度}`

调整公式编号的竖直位置,例如 `\raisetag{6pt}` 将使编号上移 6pt. 若用负的高度,则是下移位置. 通常是用正的距离,以减小公式编号行与相应公式行之间的间隔. 观察下例中公式编号与公式之间的距离,上一个减少了间距,下一个没有调整距离. 输入

```
\begin{gather}
\pi=4\Bigl(\arctan\frac{1}{2}+\arctan\frac{1}{3}\Bigr)
\doteq 3.141592653589793238462643383279\raisetag{6pt}\\
\pi=16\arctan\frac{1}{5}-4\arctan\frac{1}{239}
\doteq 3.141592653589793238462643383279
\end{gather}
```

输出

$$\pi = 4 \left(\arctan \frac{1}{2} + \arctan \frac{1}{3} \right) \doteq 3.141592653589793238462643383279 \quad (9.18)$$

$$\pi = 16 \arctan \frac{1}{5} - 4 \arctan \frac{1}{239} \doteq 3.141592653589793238462643383279 \quad (9.19)$$

9.15.4 特殊的上下标 (上下限)

命令

```
\sideset{左侧角标}{右侧角标}主体符号
```

用于在主体符号的两侧放置上下标。

命令

```
\overset{上标}主体符号 \underset{下标}主体符号
```

分别用于在主体符号的上方或下方放置上标或下标。例如输入

```
\[ \sideset{^a\_tag}{^*_b}\prod\qqquad
\overset{abc}{XY}\qqquad \underset{*}{Z} \]
```

输出为

$$^a\prod_b^* \quad \overset{abc}{XY} \quad \underset{*}{Z}$$

下列几个命令

```
\overleftarrow{表达式} \underleftarrow{表达式}
\overrightarrow{表达式} \underrightarrow{表达式}
\overleftrightharrow{表达式} \underleftrightharrow{表达式}
```

用于在表达式的上方或下方放置水平箭头,命令的名字已表明了箭头的方向和上下位置,箭头长短会根据表达式的宽度自动调整.例如

```
\[ \overlefttrightarrow{ABCDE} =
      \underrightarrow{ABC} + \overleftarrow{uvwxyz}
\]
```

输出为

$$\overleftarrow{ABCDE} = \underline{ABC} + \overleftarrow{uvwxyz}$$

当上述箭头出现在角标位置时,会自动使用较小的尺寸.
命令

```
\xleftarrow[下方表达式]{上方表达式}
\xrightarrow[下方表达式]{上方表达式}
```

在上下居中的位置画出水平箭头,并把上方表达式放置在箭头的上方.若有可选项下方表达式,就把它放在箭头的下方.箭头的长度会自动伸缩以适合表达式的长度.上下表达式自动使用角标尺寸.输入

```
\[ A\xleftarrow{1,2\,\text{行交换}} B
      \xrightarrow{II-I\times 4} C \]
```

输出为

$$A \xleftarrow{1,2 \text{ 行交换}} B \xrightarrow{II-I \times 4} C$$

9.15.5 不可断行的区间符

文章中有时用到区间符,例如语句“调查10–15岁的学生”中的“–”就是区间符,用于表示一个范围.为了避免在区间符与随后数字之间可能产生的换行,应在区间符前加上命令\nobreakdash,例如输入成10\nobreakdash--15.

第十章 新字体选择方案 (NFSS)

为了容易地选择和使用各种字体, Frank Mittelbach 和 Rainer Schöpf 于 1989 年为 L^AT_EX 提出了一个新字体选择方案——NFSS (New Font Selection Scheme), 并在 1990 年初给出了一个初步的测试软件包, 到 1993 年中, 第二版问世. 1994 年 4 月正式发行的 L^AT_EX 2_ε 将 NFSS 作为重要标准. 在基础篇的 § 3.1 中曾对 NFSS 作了很简单的介绍.

§ 10.1 NFSS 中的字体属性

在 NFSS 中, 字符集按如下 5 种属性分类: 编码 (encoding)、族 (family)、系列 (series)、形状 (shape) 和尺寸 (size), 选择这些属性的不同组合就选择了不同的字符集. 可以使用下面命令选择这些属性:

`\fontencoding{编码}`

`\fontfamily{族}`

`\fontseries{权_宽度}`

`\fontshape{形状}`

`\fontsize{字体尺寸}{行距}`

编码属性定义了字体中字符的布局, 它的一些可能值列在表 10.1 中. 通常在一篇文档中不大可能改变编码, 除非要激活其他特殊的字体如西里尔 (Cyrillic) 字体.

表 10.1 NFSS 的编码

编码	描述	字体样例	页码
OT1	来自于 Knuth 的原来的文本字体	cmr10	371
OT2	Washington 大学的 Cyrillic 字体	wncyr10	376
T1	Cork(DC/EC)字体	ecrm1000	377
OML	T _E X 数学字母字体	cmmi10	374
OMS	T _E X 数学符号字体	cmsy10	374
OMX	T _E X 数学扩展字体	cmex10	375
U	未知编码	-	

在 `\fontfamily` 命令中的族参数值表示字体的一组基本属性. 对于 CM 字体 (计算机现代字体——Computer Modern font), 族 `cmr` 包含所有 serif (衬线) 字体,

族 `cmss` 包含所有 sans serif (无衬线) 字体, 族 `cmtt` 包含 typewriter (打字机) 字体, 一些特殊装饰性字体是它们所在族的唯一成员. 在表 10.3 中根据族和其他属性列出了所有的 CM 字体.

注意: 字体属性“族”与原来 $\text{T}_\text{E}\text{X}$ 中的同名概念之间没有任何关系. $\text{T}_\text{E}\text{X}$ 中的族是由在数学公式中做为普通文本、角标和二级角标所用的 3 种不同尺寸的字体组成.

在 `\fontseries` 中的参数值 `权_宽度` 表示字符的权 (笔画的粗细程度) 和宽度 (字符的宽紧程度). 它们是由表 10.2 中所给的 1 到 4 个字母来定义. 表中的“粗”也可称为“黑”.

表 10.2 NFSS 的系列属性

权类			宽度类		
极细	Ultralight	ul	极紧	50%	uc
特细	Extralight	el	特紧	62.5%	ec
细	Light	l	紧 (Condensed)	75%	c
半细	Semilight	sl	半紧	87.5%	sc
中等 (正常)	Medium (Normal)	m	中等 (正常)	100%	m
半粗	Semibold	sb	半松	112.5%	sx
粗	Bold	b	松 (Expanded)	125%	x
特粗	Extrabold	eb	特松	150%	ex
极粗	Ultrabold	ub	极松	200%	ux

`\fontseries{权_宽度}` 的参数值是由对应于权的字母后接对应于宽度的字母组成. 因此 `ebsc` 表示权为特粗, 宽度为半紧, 而 `bx` 意味着权为粗, 宽度为松. 同任何非中等 (正常) 权或宽度组合时, 可以省略字母 `m`; 如果两者都是中等 (正常) 值, 则只要给出一个 `m` 就行了.

在 `\fontshape` 中, 参数值 `形状` 是 `n`, `it`, `sl` 或 `sc` 字母组合中的一种, 分别表示正常 (直立)、*italic* 斜体、*slanted* 斜体或小体大写字母.

`\fontsize` 属性命令有两个参数值, 第一个参数值 `字体尺寸` 表示字体的大小, 第二个参数值 `行距` 表示选择该字体后上下两行基线之间的垂直距离, 不显式地给出单位时都以 `pt` 为单位. 选择这个属性后, 第二个参数值就成为行距 `\baselineskip` 的新值. 例如, `\fontsize{12}{15}` 选择 12pt 的字体尺寸, 行距为 15pt.

设置一个或几个属性后, 就可以紧跟着用 `\selectfont` 命令激活具有当前属性的字体. 各种属性是彼此独立的, 改变其中一个, 并不会改变另一个. 例如

```
\fontfamily{cmr} \fontseries{bx} \fontshape{n}
\fontsize{12}{15} \selectfont
```

就选择了一种直立、粗体、松的罗马字体, 尺寸为 12pt, 行距 15pt. 如果后来又用 `\fontfamily{cmss}` 设置一种无衬线字体, 那么在进行下一次 `\selectfont` 调用时, 属性中的权和宽度 `bx`、形状 `n`、尺寸 12(15)pt 继续有效.

等价地, 可以利用下面这条命令来定义除尺寸外的所有属性, 并同时马上激活字体:

`\usefont{编码}{族}{系列}{形状}`

由 F. Mittelbach 和 R. Schopf 提供的表格 10.3, 列出了根据族、系列和形状属性对计算机现代字符集的分类. 有相当多的属性组合并不对应某个 CM 字体, 这不是 NFSS 系统的缺陷, 这一设计是为将来考虑的, 它有充分的发展余地. 它也可以应用于正变得越来越普及的 PostScript 字体.

表 10.3 计算机现代字体的属性

系列	形状	外部字体名称示例
计算机现代罗马字体 (<code>\fontfamily{cmr}</code>)		
m	n, it, sl, sc, u	cmr10, cmti10, cmsl10, cmcsc10, cmu10
bx	n, it, sl	cmbx10, cmbxti10, cmbxsl10
b	n	cmb10
计算机现代无衬线字体 (<code>\fontfamily{cmss}</code>)		
m	n, sl	cmss10, cmssi10
bx	n	cmssbx10
sbc	n	cmssdc10
计算机现代打字机字体 (<code>\fontfamily{cmtt}</code>)		
m	n, it, sl, sc	cmtt10, cmitt10, cmsl10, cmtcsc10

形式上是可以任意设置属性的组合的, 然而, 可能不存在一种实际字体对应于所有选择的属性. 如果出现这种情况, 那么当调用 `\selectfont` 时, \LaTeX 会给出一条警告信息, 告诉你它用什么字体取代所需字体. 在 `\fontsize` 命令中的字体尺寸属性通常可以取 5、7、8、9、10、10.95、12、14.4、17.28、20.74, 但也可以是其他值. 第二个参数值, 即基线间距, 可以取任何值, 因为它并不是字体固有的性质.

利用 `\begin{document}` 命令, \LaTeX 给 5 种属性设置当前特定默认值. 这通常就是标准编码 OT1, 族 `cmr`, 中等 (正常) 系列 `m`, 正常形状 `n` 和在文档类选项中指定的基本尺寸. 用户可以在导言中改变这些值, 或者用特殊选项把它们设置成不同的值.

§ 10.2 简化的字体选择命令

属性命令 `\fontencoding`、`\fontfamily`、`\fontseries`、`\fontshape` 和 `\fontsize`, 连同命令 `\selectfont`, 都是新字体选择方案中的基本工具. 用户并不

需要直接使用这些命令, 而可以利用列在 § 3.1 中的命令和声明. 类似于 `\itshape` 这样的字体声明实际定义就是

```
\fontshape{it}\selectfont.
```

族声明 (及对应的标准族属性值) 为

```
\rmfamily (cmr) \sfamily (cmss) \ttfamily (cmtt)
```

这分别对应于计算机现代字体中的罗马、无衬线和打字机字体.

系列声明 (及其初始值) 为

```
\mdseries (m) \bfseries (bx)
```

这表明在标准中只提供了正常和粗松两种系列.

形状声明 (及相应属性值) 为

```
\upshape (n) \itshape (it) \slshape (sl) \scshape (sc)
```

这可以选择直立、斜体、slanted 和小体大写字母.

利用 `\normalfont` 命令, 可以随时把族、形状和系列属性值重设为标准值, 但不改变字体尺寸.

对于上面每种字体属性声明, 也存在着一个相应的字体命令, 用以设置其参数文本的字体. 因此 `\textit{text}` 就与 `{\itshape text}` 差不多一样, 区别在于命令中自动包含倾斜校正, 并且命令中的文本参数不能位于两段. 这些命令的清单请参见 § 3.1.

注意对编码并没有类似的声明命令, 这是因为通常并不需要在文档中改变编码. 如果要改变编码, 例如要用西里尔 (cyrillic) 字体时, 需要用到编码 OT2, 可以进行如下定义:

```
\newcommand{\cyr}{\fontencoding{OT2}\selectfont
\renewcommand{\bfseries}{\fontseries{b}\selectfont}}
\newcommand{\lat}{\fontencoding{OT1}\selectfont}
```

这样就可以方便地在 OT1 和 OT2 编码之间来回切换了. 其中的命令 `\cyr` 和 `\lat` 可以由用户按个人喜好改用其他字符串. 不过为了使用俄文黑体 (`\bfseries`), 应该要把黑体命令放在 `\cyr` 生效的范围内, 例如应该用 `{\cyr\textbf{...}}`, 不要用 `\textbf{\cyr ...}`.

下面以正文中出现俄文作为改变编码的例子. 有时文章中可能包含一些俄文或其他斯拉夫语言, 这就需要使用西里尔字母, 因此要把当前编码改变成 OT2 编码. 不过在 MiKTeX 的基本安装中并不包括西里尔字母, 如果执行 \TeX 出现缺少文件的错误, 就要补充安装 Languages → Cyrillic 模块. 在 OT2 中, 已定义了很多字符编码, 使得直接输入英文字母就会输出相应的西里尔字母, 例如文稿中有俄文单词 алгоритм, 可通过输入 `{\cyr algorifm}` 得到. 下表是一些俄文字母与英文字母的对应表, 因前者个数多于后者, 所以有些俄文字母需输入成英文字母的组合. 表中只有小写的俄文字母, 需要大写字母时只要相应的输入大写英文字母就行了, 若是

英文字母组合, 只需第一个字母大写. 从表中看出, 俄文字母基本上是按发音输入相应的英文字母:

а	а	б	б	в	в	г	г	д	д	е	е
ё	\ "e	ж	zh	з	z	и	i	й	\ U i	к	k
л	l	м	m	н	n	о	o	п	p	р	r
с	s	т	t	у	u	ф	f	х	kh	ц	ts
ч	ch	ш	sh	щ	shch	ъ	\ cyrrhdsn			ы	y
ь	\ cyrsftsn			э	\ cyrrev			ю	yu	я	ya

注意输入 {\cyr ts} 产生 ц, 如果输入 {\cyr {t}s} 则产生 тс. 欲知其中详情, 请参见 MiKTeX 的 \texmf\doc\latex\cyrillic 目录内的说明文件.

华盛顿大学设计的西里尔字体名称都以 wncy 开头, 后接样式标志 r (直立)、b (粗体)、i (斜体)、sc (小体大写) 或者 ss (sans serif 字体), 然后是设计尺寸的点数.

与编码有关的一些字体描述文件 (扩展名为 fd 的文件) 将属性组合与具体的字体名称关联在一起, 使得我们也能使用各种不同的西里尔字体. 例如

{\cyr\textrm{ABCD abcd 1234}}	АБЦД абцд 1234
{\cyr\textbf{ABCD abcd 1234}}	АБЦД абцд 1234
{\cyr\textit{ABCD abcd 1234}}	<i>АБЦД абцд 1234</i>
{\cyr\textsc{ABCD abcd 1234}}	АБЦД АБЦД 1234
{\cyr\textsf{ABCD abcd 1234}}	АБЦД абцд 1234
{\tiny\cyr ABCD abcd 1234}	АБЦД абцд 1234

§ 10.3 属性的默认值

在前面一节中的字体属性声明并没有显式地设置属性的值, 而是用某种默认的命令来进行. 例如 \itshape 的真正定义是:

```
\fontshape{\itdefault}\selectfont
```

可用的默认命令有:

族: \rmdefault \sfdefault \ttdefault

系列: \mddefault \bfdefault

形状: \updefault \itdefault \sldefault \scdefault

当调用了 \normalfont 命令时, 需要定义标准属性值. 它们是由如下 4 个默认值组成的:

\encodingdefault	\familydefault
\seriesdefault	\shapedefault

默认值应由程序设计者用低级命令去定义它们, 用户只需知道对于 OT1 编码有 3 个族、2 个系列和 4 种形状的标准属性值可供直接使用.

§ 10.4 定义新的字体命令

有很多命令可以用来定义新的字体声明和命令, 这些命令主要是为 L^AT_EX 宏包开发者服务的, 但也可以用在普通文档中.

```
\DeclareFixedFont{\命令}{编码}{族}{系列}{形状}{尺寸}
```

把 `\命令` 定义为一个选择具有指定属性字体的声明. 所有属性都是严格固定的. 这与 `\newfont` 基本等价, 除了这里的字体是由属性而不是由名称确定的.

```
\DeclareTextFontCommand{\命令}{字体指定}
```

定义 `\命令` 为一个字体命令, 它按照字体指定设置其参数值. 在内部就是用这条命令来定义所有类似于 `\textbf` 的命令, 而定义 `\textbf` 时字体指定为 `\bfseries`.

```
\DeclareOldFontCommand{\命令}{文本指定}{数学指定}
```

定义 `\命令` 为按照 L^AT_EX 2.09 方式可以用在数学模式中的字体声明. 例如, `\it` 的定义为

```
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
```

注意: 此处定义的 `\命令` 是一个声明, 该条语句对于定义与原来版本兼容命令是相当有用的, 但应尽量避免使用.

§ 10.5 定义新的数学字体命令

10.5.1 数学字母字体

在 § 4.14.1 介绍了可以在数学模式内部调用的数学字体命令有

```
\mathrm \mathit \mathtt \mathsf \mathbf \mathcal
```

这些命令开始字符都是 `math`, 其后的字母表明了字体形状. 其中最后一个命令 `\mathcal` 是数学花体, 它们把作为命令参数的字母设置成特定的字体.

用户可以定义新的数学字体命令, 使字母表中的字母在数学模式中具有新的字体. 命令为

```
\DeclareMathAlphabet{\命令}{编码}{族}{系列}{形状}
```

上面提到的数学字体没有 *slanted* 斜体, 我们可以定义这种数学斜体, 为此需要定义一个新的数学字体命令 (必须放在导言区):

```
\DeclareMathAlphabet{\mathsl}{OT1}{cmss}{m}{sl}
```

这意味着新定义的数学字体命令 `\mathsl` 选择编码为 OT1, 族为 cmss, 权与宽度都是 m 以及形状为 sl 的字体。

在 \LaTeX 中已有两种数学变体 (version, 或称为种类) 的名称, 分别是 `normal` 和 `bold`. 对于已用 `\DeclareMathAlphabet` 命令定义的命令, 若指定它在某种数学变体下不同的字体属性, 应使用命令 (也要放在导言区)

```
\SetMathAlphabet{\命令}{变体}{编码}{族}{系列}{形状}
```

例如若指定新定义的命令 `\mathsl` 在变体 `bold` 中的系列属性变为 `bx`, 可在导言区放上命令

```
\SetMathAlphabet{\mathsl}{bold}{OT1}{cmss}{bx}{sl}
```

下面看一看新字体命令的输出结果 (逗号前是默认的数学字体):

```
$ ABxy, \mathsl{ABxy} $ ABxy, ABxy
\boldmath $ ABxy, \mathsl{ABxy} $ \unboldmath ABxy, ABxy
```

在 § 9.1 中提到国际标准中用无衬线斜体表示矩阵与 2 阶张量, 但没有这种现成的数学字体命令, 因此定义了一个新的数学字体命令 (必须放在导言区):

```
\DeclareMathAlphabet{\mathsfsl}{OT1}{cmss}{m}{sl}
```

若指定它在数学变体 `bold` 下不同的字体属性, 可用命令 (要放在导言区)

```
\SetMathAlphabet{\mathsfsl}{bold}{OT1}{cmss}{bx}{sl}
```

但从表 10.3 中看到目前还没有一种实际的字体具有上述的属性组合, 因此实际选择的是近似的字体——形状为 n. 输出形如 **ABxy**.

10.5.2 数学符号字体

在 NFSS 下可用下述命令定义一个符号字体名称

```
\DeclareSymbolFont{符号字体名称}{编码}{族}{系列}{形状}
```

它使符号字体名称与指定的属性组合联系起来. 通过这个名称可以定义具体的数学符号.

在标准的 \LaTeX 中已定义了如下几个符号字体名称:

```
\DeclareSymbolFont{operators}{OT1}{cmr}{m}{n}
\DeclareSymbolFont{letters}{OML}{cmm}{m}{it}
\DeclareSymbolFont{symbols}{OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
```

使用已定义的符号字体名称, 可以构造数学字母表和各种不同类型的符号.

```
\DeclareSymbolFontAlphabet{\数学字母表}{符号字体名称}
```

将数学字母表定义成基于符号字体名称的数学字母表. 如在导言区放上命令

```
\DeclareSymbolFontAlphabet{\testi}{letters}
```

那么输入 $\text{\testi{Ax}}$ 就会输出 Ax , 这就是符号字体名称 `letters` 对应的属性组合中位于字母表 A、x 位置的字符. 如果在导言区放的命令是

```
\DeclareSymbolFontAlphabet{\testii}{symbols}
```

则输入 $\text{\testii{Ax}}$ 就会输出 $A\mathfrak{s}$, 它们是符号字体名称 `symbols` 对应的属性组合中位于字母表 A、x 位置 (即第 65 和 120 号位置) 的字符.

定义符号的主要命令是

```
\DeclareMathSymbol{\符号}{类型}{符号字体名称}{位置}
```

此后符号就代表字母表中位于指定位置的字符, 字母表中的字体具有符号字体名称对应的属性组合. 命令中的位置是一个整数, 可以用十进制 (如 65), 也可以用八进制 (如 '81), 或者用十六进制 (如 "41). 类型用于区分符号的类别和功能, 它可以取如下的一个值:

<code>\mathord</code>	普通 (normal) 符号
<code>\mathop</code>	大运算符 (large operator), 如 \sum
<code>\mathbin</code>	二元运算符 (binary operator), 如 \times
<code>\mathrel</code>	关系运算符 (relational operator), 如 \geq
<code>\mathopen</code>	左括号 (opening bracket), 如 $\{$
<code>\mathclose</code>	右括号 (closing bracket), 如 $\}$
<code>\mathpunct</code>	标点符号 (punctuation)
<code>\mathalpha</code>	字母表字符 (alphabet character)

建立数学重音的命令是

```
\DeclareMathAccent{\命令}{类型}{符号字体名称}{位置}
```

其中类型既可以是 `\mathord` 又可以是 `\mathalpha`.

具有两种尺寸的数学符号如定界符 (delimiter)、根号 (radical) 可如下定义:

```
\DeclareMathDelimiter{\命令}{类型}{符号字体名称 1}{位置 1}
{符号字体名称 2}{位置 2}
```

```
\DeclareMathRadical{\命令}{符号字体名称 1}{位置 1}
{符号字体名称 2}{位置 2}
```

当需要小尺寸符号时, \命令取符号字体名称 1 中位置 1 处的符号, 当需要大尺寸符号时, 则取符号字体名称 2 中位置 2 处的符号.

上述各个命令中没有指定尺寸属性, 这是因为数学模式需要 4 种尺寸 (参见 § 4.1 和 § 9.2). 这些尺寸是由下述命令指定的 (如要自行指定, 需放在导言区):

```
\DeclareMathSizes{正文}{数学文本}{角标}{二级角标}
```

其中各个参数都是数字, 表示以点 (pt) 为单位的字体大小. 如果当前正文尺寸 (以点为单位, 下同) 恰好等于命令中的参数 正文, 则数学模式中的

```
\textstyle \scriptstyle \scriptscriptstyle
```

就分别取参数 数学文本、角标 和 二级角标 的值. 但若当前的正文文本不等于命令参数中的 正文 数值, 则数学模式中的尺寸命令就不受任何影响.

一般来讲用户没有必要使用这条命令, 但为了容易理解命令的作用, 现举一例: 设正文标准尺寸是 11pt, 在导言区放上命令

```
\DeclareMathSizes{12}{20}{14}{10}
```

下面是不同输入产生的不同输出:

{ \large \$x^{x^x}=B_{k_j}\$ }	$x^{x^x} = B_{k_j}$
{ \Large \$x^{x^x}=B_{k_j}\$ }	$x^{x^x} = B_{k_j}$
\$x^{x^x}=B_{k_j}\$	$x^{x^x} = B_{k_j}$

这是因为 \large 将字体尺寸设置为 12pt, 与命令中的参数匹配. 而 \Large 和正常尺寸都不是 12pt, 故不受该命令的影响. 需要提醒的是不要随便使用这条命令, 否则可能会在一些意想不到的地方影响其他命令的效果, 例如 \subsection 标题默认使用 12pt 字体, 如果定义了上述命令而又在标题中使用了数学字体, 那么结果就出乎意料了. 请自行编译看看例 10-5-1.tex 的输出.

§ 10.6 在 NFSS 下指定字体

在 NSFF 下, 当在文档中指定字体时, 可以先给出所需要的属性组合, 然后调用命令 \selectfont. 这种字体属性组合与特定的外部字体名称的联系是通过字体定义命令完成的, 这些命令通常存储在扩展名为 def 和 fd 文件中. 下面对其中一些命令作简单介绍. 首先用

```
\DeclareFontEncoding{编码}{文本模式命令}{数学模式命令}
```

设置一个称为 编码 的新的编码属性, 以后当选择这种编码的文本字体时, 就会执行文本模式命令, 重定义重音命令或其他依赖于编码的事情. 类似地, 对于这种编码的数学字母就会调用数学模式命令. 也可如下定义默认的文本模式命令和数学模式命令:

```
\DeclareFontEncodingDefault{文本模式命令}{数学模式命令}
```

这条命令声明了一般的文本和数学模式设置, 更具体的设置出现在前面的命令 `\DeclareFontEncoding` 中.

对于指定的属性组合, 如果不存在实际的字体与之对应, 下述声明指定了要被取代的属性值:

```
\DeclareFontSubstitution{编码}{族}{系列}{形状}
```

取代是按着 形状、系列、族的次序进行, 编码永远不会被取代. 如果这样取代后仍找不到对应的字体, 那么就用

```
\DeclareErrorFont{编码}{族}{系列}{形状}{尺寸}
```

确定最终应使用的字体.

在特定的编码方案中建立一个新的族使用如下命令:

```
\DeclareFontFamily{编码}{族}{命令序列}
```

每当选择这种编码中该族的字体时, 就会先执行指定的命令序列.

把字体属性与外部字体名称关联起来的主要字体定义声明是

```
\DeclareFontShape{编码}{族}{系列}{形状}{字体定义}{命令序列}
```

每当选择其中的字体时, 就会先执行指定的命令序列.

字体定义由一些尺寸/字体关联项组成, 每一项包含尺寸部分、一个函数、一个可省略参数和一个字体参数. 例如

```
\DeclareFontShape{OT1}{cmr}{m}{n}
{ <5> <6> <7> <8> <9> <10> <12> gen * cmr
  <10.95> cmr10
  <14.4> cmr12
  <17.28> <20.74> <24.88> cmr17}{}
```

说明 `cmr` 族的中等系列、正常形状的成员用外部字体 `cmr5`, ..., `cmr10`, `cmr12` 表示 5-12pt 的尺寸, 以 `cmr10` 放大到 10.95pt 表示 10.95pt 或 11pt 的尺寸, 等等. 如果指定的尺寸不存在, 就用一定限度内最接近的尺寸代替.

尺寸部分放在尖括号内的数表示点数 (pt), 括号内也可以包含范围, 如 `<-10>` 表示所有小于 (但不包括) 10 pt 的尺寸, `<24->` 表示大于或等于 24 pt 的尺寸, `<10-24>` 表示大于或等于 10 pt 并且小于 24 pt 的尺寸, 相当于是数学上的左闭右开区间 $[10, 24)$.

字体定义中可以使用的函数有:

(无) (上面例中的后 3 行) 加载指明的字体, 放缩到要求的尺寸. 如果在字体名称前面有位于方括号内的选项, 则它是一个附加的放缩因子, 例如

`<11> [.95] cmr10` 加载 `cmr10` 并放缩到 11 pt 的 95%;

gen * 把点数尺寸加到字体参数后, 生成字体名称, 例如

`<5> <12> gen * cmr` 加载 `cmr5`, `cmr12`;

genb * 把点数尺寸乘以 100 加到字体参数后, 生成字体名称, 用于 DC 和 EC 字体, 例如

`<14.4> genb * ecss` 加载 `ecss1440`;

sub * 替代字体, 这种字体的属性以族/系列/形状的参数形式给出, 例如

`<-> sub * cmtt/m/n` 当没有具有所需属性的字体时, 最好使用这种替代字体; 会显示并在 `log` 文件中记录一条消息;

subf * 类似于上面的空函数, 但在加载一种替代字体时会给出警告消息;

fixed * 以正常尺寸加载指定的字体, 忽略尺寸部分; 如果给出了可选项参数, 这个参数就是字体要放缩到的尺寸, 例如

`<10> fixed * [11] cmr12` 当要求的是 10 pt 时, 加载 `cmr12` 并放缩到 11 pt.

上面所有函数都可以加一个前缀字母 `s` (表示无声), 以禁止在屏幕上显示信息, 例如 `sub *` 的无声形式是 `ssub *`, 空函数的无声形式是 `s *`.

下面再考虑一个实际的例子: 定义黑、斜打字机字体:

```
\DeclareFontShape{OT1}{cmtt}{bx}{it}{
  <-> ssub * cmtt/m/it }{}
```

因为在计算机现代字体集中没有满足属性组合 `{OT1}{cmtt}{bx}{it}` 的字体, 所以由该命令定义的字体实际将使用替代字体, 包括所有尺寸 (`<->`) 都用系列属性为 `m`、形状属性为 `it` 的打字机字体代替.

字体定义命令既可以放在宏包文件里, 也可以在文档中使用. 但通常是先把每条 `\DeclareFontEncoding` 命令存储在一个文件中, 命名为 编码`enc.def`, 例如 `OT1` 编码对应的文件名是 `ot1enc.def`, 然后把命令 `\DeclareFontFamily` 和 `\DeclareFontShape` 放在扩展名为 `fd` 的文件中, 该文件主名由编码和族名组成, 例如文件 `ot1cmr.fd`. 当选择的编码和族的组合在文档或调用的宏包中没有定义时, `LATEX` 就会自动查找相应的 `fd` 文件并把它调入, 因此无需显式输入这种文件.

但要注意, 如果在当前的格式中不知道编码, 就必须显式地或通过加载 编码`enc.def` 文件调用 `\DeclareFontEncoding`. 加载这种文件的一个方法是调用已有的宏包 `fontenc`, 例如

```
\usepackage[OT2,T1]{fontenc}
```

需要的编码作为选项放在方括号中, 其中最后一个就被置为当前编码.

下面以在 NFSS 下安装 PostScript 字体作为一个例子, 一些常用的 PostScript 字体已经在它们自己的 `fd` 文件中作为单独的族定义好了, 例如, `ot1ptm.fd` 就把

PostScript 的 Times 字体与一个名为 ptm 的族关联在一起. 激活这些字体的宏包在文件 times.sty 中, 其中包括重要的如下几行:

```
\renewcommand{\rmdefault}{ptm}
\renewcommand{\sfdefault}{phv}
\renewcommand{\ttdefault}{pcr}
```

这使得 Times ptm 成为默认的罗马字体族, 用 \rmfamily 调用; Helvetica phv 成为默认的无衬线字体族, 用 \sfdefault 调用; Courier pcr 成为默认的打字机字体族, 用 \ttdefault 调用.

下面 3 行是分别使用命令 \textrm、\textsf 和 \texttt 的输出结果, 左边是通常使用的字体, 右边是如上设置的 PostScript 字体.

ABCDEFabcdef123456	ABCDEFabcdef123456
ABCDEFabcdef123456	ABCDEFabcdef123456
ABCDEFabcdef123456	ABCDEFabcdef123456

§ 10.7 编码命令

在 L^AT_EX 中需要使用命令来得到特殊的字符和重音, 例如用 \O 显示斯堪地纳维亚字符 Ø. 这个字符在字体表中的位置与编码有关, 它在 OT1 中是第 31 个字符, 而在 T1 中是第 216 个字符. 因此当编码改变了时, 需要重新定义所有这样的符号命令. 这可借助于某种编码命令来完成, 这些命令通常与 \DeclareFontEncoding 命令一同位于 编码 enc.def 文件中.

为了定义一个在不同编码下功能不同的命令, 可以使用

```
\ProvideTextCommand{\命令}{编码}[参数个数][可选项]{定义}
\DeclareTextCommand{\命令}{编码}[参数个数][可选项]{定义}
```

它们的行为同 \providecommand 和 \newcommand 命令一样, 只是这样定义的 \命令 只有当 编码 被激活时才具有这里给出的定义. 因此对每种编码, \命令 具有不同的定义.

```
\DeclareTextSymbol{\命令}{编码}{位置}
```

定义 \命令 为当 编码 被激活时在字体表中给定 位置 处的字符.

```
\DeclareTextAccent{\命令}{编码}{位置}
```

定义 \命令 为一个重音符号, 当 编码 被激活时使用字体表中给定 位置 处的字符作为重音符号.

```
\DeclareTextCompositeCommand{\命令}{编码}{字母}{定义}
```


定义 \命令 及后接的单个字母的行为是执行定义。

```
\DeclareTextComposite{\命令}{编码}{字母}{位置}
```

定义 \命令 及后接的单个字母的行为是显示字体表中给定位置处的字符。这个声明在 T1 编码中很有用，因为在 T1 编码中许多重音字母就是一个单独的符号，例如第 233 个字符就是 é，所以在 T1 编码中可先作声明

```
\DeclareTextComposite{\'}{T1}{e}{233}
```

然后用 \'e 得到 é。在 OT1 编码中，这个重音符号是一个单独符号，位于第 19 个位置，因此在 OT1 编码中应先作声明

```
\DeclareTextAccent{\'}{OT1}{19}
```

然后用 \'e 得到 é。

上述的具体声明已包含在文件 t1enc.def 和 ot1enc.def 中，只须激活相应的编码就可直接使用重音命令了。

对于未声明的编码命令可以给出默认定义：

```
\DeclareTextCommandDefault{\命令}[参数个数][可选项]{定义}
\ProvideTextCommandDefault{\命令}[参数个数][可选项]{定义}
\DeclareTextAccentDefault{\命令}{编码}
\DeclareTextSymbolDefault{\命令}{编码}
```

前两条命令给出了适用于所有编码的 \命令 的默认定义，后两条则指定 \命令 把给定的编码作为默认值。

上述一些命令是于 1994 年 12 月 1 日加到 L^AT_EX 2_ε 中的，因此在使用这些命令的文件中应包含

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
```

§ 10.8 计算机现代字体

10.8.1 简介

当初 Donald E. Knuth 设计 T_EX 程序时，同时设计了丰富的字符集或字体，称它们为计算机现代字体 (CM 字体)，后来人们在不断地设计增加新的 CM 字体，这些字体被分成了几个族。由于它们是在 NFSS 属性分类系统建立之前出现的，CM 字体命名法并不与 NFSS 完全一致。NFSS 使得用户不必记住 CM 字体名称的细节，而只需指定属性。但是使用 NFSS 必须有字体描述文件 (扩展名为 fd 的文件)，这个文件把属性组合转化为实际的字体名称。

所有 T_EX 字体文件的名称都以 cm (表示 “Computer Modern”) 开头, 后接一至四个字母描述字体的样式, 最后是一或两个数字指定设计尺寸的点数, 扩展名是 tfm, 意为 “T_EX font metric”.

在 tfm 文件中并没有包含生成符号的信息, 而是由字体尺寸信息组成, 这些尺寸包括宽度、高度和深度. 对于 *slanted* 字体, 每个字母还有倾斜校正. 此外, 要为一些字母组合指定与通常不同的间距, 如 AV 与 $\mathcal{A}\mathcal{V}$ 的区别, 以及对一些字母进行连写的信息. 还包括字符的斜率 (可以为零)、单词间距及其伸展和收缩量、em 和 quad 间距的大小、句子结尾处的间距. 对于数学和符号字体, 还包括更多的信息.

在安装 L^AT_EX 时已安装了很多 CM 字体, 要想了解究竟有哪些字体, 只需在 T_EX 系统中查找扩展名为 tfm 的文件. 下表是部分例子 (仅以 10 点为例):

字母	含义	字体名示例
r	直立罗马字体	cmr10
sl	<i>slanted</i> 字体 (倾斜了 1/6 的罗马字体)	cmsl10
csc	小体大写直立罗马字体	cmcsc10
ti	文字模式 <i>italic</i> 字体	cmti10
u	‘扶直’了的 <i>italic</i> 字体	cmu10
b	普通粗体	cmb10
bx	宽松粗体	cmbx10
bxsl	粗斜体	cmbxsl10
bxti	粗斜体	cmbxti
ss	直立无衬线字体	cmss10
ssi	倾斜无衬线字体	cmssi10
ssbx	无衬线宽松粗体	cmssbx10
tt	直立打字机字体	cmtt10
tcsc	小体大写打字机字体	cmtcsc10
sltt	倾斜打字机字体	cmsl10
itt	倾斜打字机字体	cmitt10
mi	数学斜体	cmmi10
mib	数学斜粗体	cmmib10
sy	数学符号	cmsy10
bsy	粗体数学符号	cmbsy10

10.8.2 使用 CM 字体

L^AT_EX 提供的现成的字体命令只有有限的几个, 如果用户需要一些特殊的尺寸, 可通过放大或缩小已安装的字体来达到目的. 命令

```
\newfont{\名字}{字体 at 尺寸}
\newfont{\名字}{字体 scaled 因子}
```

定义了一个新的字体名称\名字. 命令中的字体是已有字体的外部文件的主名. 第一种命令把已有字体放缩到指定的尺寸作为自定义字体的大小, 第二种命令中的因子除以 1000 后的值是真正的放缩因子, 把已有字体的尺寸乘以放缩因子得到的值就是自定义字体的大小. 例如需要 123.45pt 的直立罗马字体, 可如下定义

```
\newfont{\myfnt}{cmr10 at 123.45pt}
```

或

```
\newfont{\myfnt}{cmr10 scaled 12345}
```

以后\myfnt就可作为字体尺寸声明使用了, 就像使用\tiny、\huge等声明一样. 但要注意, 这种自定义的字体仅仅具有字体的大小, 而不改变所在段落的行距(基线间距)\baselineskip的值. 如果定义新的字体名称时不带at或scaled参数, 则使用字体原有的尺寸. 通过上述两个命令定义新字体命令, 就可以使用所有已安装的字体了.

在某些T_EX环境中, 可能没有安装OT2编码, 或没有按NFSS格式安装OT2编码, 因此不能像前面介绍的那样使用西里尔字母. 此时可以像上面介绍的那样通过定义新的字体命令来使用西里尔字体, 例如

```
\newfont{\wncy}{wncyr10}
```

定义了一个新的字体声明, 激活 10 pt 的西里尔字体. 使用这个声明后, 输入英文字母就会输出相应的西里尔字母, 例如 алгоритм 可通过输入 {\wncy algorifm} 得到.

每种字体集内的每个字符都有一个序号确定它在该字符集内的位置, 利用序号也可以得到当前字体集内对应的字符. 命令为

```
\symbol{序号}
```

其中序号可以是十进制, 或是八进制(数字前带前缀“\”), 或十六进制(数字前带前缀“\”). 例如, 为了排版输出倒引号“`”, 直接使用键盘上的倒引号不能达到目的, 因为它的输出是英文左单引号“'”, 实际应输入\symbol{18}. 又如表示空格“ ”的符号是打字机字体集中的 32 号字符, 可输入成下列 3 种形式之一:

```
\texttt{\symbol{32}}
\texttt{\symbol{'40}}
\texttt{\symbol{"20}}
```

顺便提及, 在L^AT_EX中使用\textvisiblespace或\verb*| |命令也可输出空格符号, 分别形如 和 .

§ 10.9 ttfshape 简介

孙文昌编写的软件包 ttfshape 提供 True Type、Type1、Type42 字体的L^AT_EX接口, 支持CCT、CJK两种中文方式, 并允许改变形状, 还提供了TTF、type1、type42矢量字体的空心Hollow Letters和加粗命令.

随书光盘已包含了该软件包, 可直接使用其中提供的命令, 需要了解该软件包详细内容的读者请阅读该包作者写的使用说明, 见安装完成后硬盘上的文件

```
\texmf\doc\latex\ttf-MiKTeX\ttfshape\ttfshape-man.pdf
```

本节仅简单介绍其中一些命令.

使用该软件包需要在导言区写上

```
\usepackage[选项]{tffshape}
```

选项为

[mathtimes] 使用 Times New Roman 作为缺省数学字体.
[upgreek] 使用直立希腊字母 (在原基础上向左倾斜 16.7%).

tffshape 预定义了以下字体可直接使用

\TTFsongti	文鼎PL简报宋
\TTFkaiti	文鼎PL简中楷
\TTFqtimes	quassi times
\TTFqbookman	quassi bookman
\TTFqcourier	quassi courier
\TTFqswiss	quassi swiss
\TTFqswisscondensed	quassi swiss condensed
\TTFqpalatino	quassi palatino

10.9.1 改变字形

四个字形宏命令:

```
\TTFslant{倾斜比}
\TTFextend{增宽比}
\TTFheighten{增高比}
\TTFsetshape{倾斜比}{增宽比}{增高比}
```

所有参数都是整数(百分比), 倾斜比的范围是 $-99 \sim 99$, 负值表示向左倾斜, 缺省值为 0; 增宽比和增高比的缺省值都是 100; 第四个命令相当于前三个命令的组合.

10.9.2 pdf \LaTeX 、dvipdfmx、dvips 补充命令

宏	\TTFhollow[dim]{texts}
适用范围	Yap / DVIPS / DVIPDFMx / PDF \LaTeX
作用	空心字体 Hollow letters

宏	<code>\TTFbold[dim]{texts}</code>
适用范围	DVIPDFMx / PDFL ^A T _E X
作用	伪黑体 (Fake bold)

宏命令 `\bfseries` 将使得中文字体以 TTFbold 方式显示, 而英文字体则使用标准黑体.

`\TTFbold` 和 `\TTFhollow` 都有一个可选参数, 以控制笔画宽度, 缺省值分别为 0.5 和 0.3.

宏	<code>\embedCJKTTF</code> <code>\notembedCJKTTF</code>
适用范围	DVIPDFMx
作用	嵌入/不嵌入 CJK TTF 字体 (缺省). 后者得到的 pdf 文件较小, 但在没有预装所需字体的机器上预览时将用其他字体 (可能是宋体, 取决于系统设置的默认替代字体) 代替.

宏	<code>\embedTTFfamily{family-name}</code> <code>\notembedTTFfamily{family-name}</code>
适用范围	DVIPDFMx
作用	嵌入/不嵌入 family-name 所对应 CJK TTF 字体.

宏	<code>\embedTTFfamilyascii{family-name}</code> <code>\notembedTTFfamilyascii{family-name}</code>
适用范围	DVIPDFMx
作用	嵌入/不嵌入 family-name 所对应 CJK TTF 字体中的英文字母.

以上三组命令的优先级由低到高, 其中

`\embedTTFCJK`, `\notembedTTFCJK` 对所有 CJK TTF 字体有效,

`\embedTTFfamily`, `\notembedTTFfamily` 仅对特定的 CJK family 有效,

`\embedTTFfamilyascii`, `\notembedTTFfamilyascii` 仅对特定的 CJK family-name 所对应的 TTF 字库中的英文字母有效.

宏	<code>\DeclareEncodingCmap{CJKenc}{Cmap}</code>
适用范围	DVIPDFMx
作用	定义 CJKenc 所用的 cmap. <code>ttshape</code> 已定义如下 Cmap: <code>\DeclareEncodingCmap{GB}{UniGB-UTF16-H}</code> <code>\DeclareEncodingCmap{GBK}{UniGB-UTF16-H}</code>

宏	<code>\TTFreplace{dest-family}</code> <code>{Regularname}{Italicname}{Boldname}{BoldItalicname}</code>
适用范围	DVIPS / DVIPDFMx
作用	用PSname分别为 <code>{Regularname}{Italicname}{Boldname}{BoldItalicname}</code> 的 Adobe 标准字体代替 dest-family 写入 ps/pdf 文件,但 不嵌入字体文件.

注意: 第 2-5 参数必须是 Adobe 定义的标准字体. 例如, 若使用 Times New Roman 和 Courier 分别替代 qtimes 和 qcourier, 相应命令 (已经由 ttfsshape 定义) 为

```
\TTFreplace{qtimes}{Times-Roman}{Times-Italic}
                        {Times-Bold}{Times-BoldItalic}
\TTFreplace{qcourier}{Courier}{Courier-Oblique}
                        {Courier-Bold}{Courier-BoldOblique}
```

注意, 这里替代与被替代字体的 tfm 必须匹配.

宏	<code>\couriertt</code>
适用范围	DVIPS / DVIPDFMx / PDFL ^A T _E X
作用	ttfamily 使用 Courier 字体.

10.9.3 直立希腊字母

利用 ttfsshape 可以旋转 ttf、t1、t42 字体的功能, 把斜体希腊字母向左倾斜一定角度即得到直立希腊字母. 使用软件包 ttfsshape 时带上选项 [upgreek], 则使所有斜体小写希腊字母向左倾斜 17% 变为直立. 若偶尔需要直立的小写希腊字母, 则不要使用选项 [upgreek]. 建议使用本书提供的宏包 greekup, 此时只要在希腊字母命令上添加 up, 就会产生直立的希腊字母, 例如 π 得到直立的 π .

第十一章 文档的布局及相互联系

在第五章中介绍了一些常用的文档类别与版式,下面再补充一些与版面布局有关的选项与命令.

§ 11.1 分栏

在 `\documentclass` 的可选项中与分栏有关的选项有两个:

<code>onecolumn</code> <code>twocolumn</code>

默认值是 `onecolumn`, 表示每页只有一栏, 即不分栏. 选项 `twocolumn` 使文稿按两栏方式排版, 两栏之间的间距以及两栏之间的分隔线的粗细都有默认值, 用户也可自行修改. 设置栏间距的命令格式是

<code>\setlength{\columnsep}{宽度}</code>

例如命令 `\setlength{\columnsep}{3em}` 设置栏间距是当前基本字体宽度的 3 倍. 设置栏间分隔线的命令格式是

<code>\setlength{\columnseprule}{宽度}</code>

例如 `\setlength{\columnseprule}{1.5pt}`, 设置栏间分隔线的宽度是 1.5pt. 默认的宽度是零, 即不显示分隔线.

表示栏宽的参数是

<code>\columnwidth</code>

对于单栏格式, 它与 `\textwidth` 相同. 对于两栏格式, 其值由 `\textwidth` 和 `\columnsep` 所决定. 用户不要自行设置栏宽值, 但可使用这个值.

上述设置命令如果放在导言区, 其值适用于整个文档, 如果放在正文主体部分, 则只有局部作用, 即其作用终止于下一次的修改, 或者终止于它所在环境或分组的结束.

如果只需要部分页面使用两栏格式, 则不能在文档类别命令中使用两栏选项, 而应在正文中使用命令

```
\twocolumn[通栏文本]
\onecolumn
```

命令 `\twocolumn` 终止当前页, 开始新的一页, 在新页里以双栏格式输出后继文本, 直到遇到 `\onecolumn` 时结束. 如果存在可选项 通栏文本, 则在新页的顶部通栏排印通栏文本的内容.

双栏排版时, 总是在排满一页中左边一栏后再排右边一栏, 这意味着当内容较少时, 只能看到一栏而不是两栏, 即不会将内容平均分成两半排成等高的两栏.

不管当前页是否为两栏格式, 命令 `\onecolumn` 总是中止当前页, 开始新的一页, 在新页里以单栏格式输出.

Sigitas Tolušis 编写了工具集, 增强了 L^AT_EX 的排版功能, 使用该工具集中的宏包 `flushend` 和 `cuted`, 可以显著改进分栏排版的效果: 使用文档类选项 `twocolumn` 后, 分栏时不再另起一页, 并且在同一页中可以既有双栏又有单栏, 而且双栏的最后一页上左右两栏基本是同高的.

首先在导言区写上

```
\usepackage{flushend, cuted}
```

则使用双栏文档选项后, 文档内容就会被平均排成两栏. 如果仍要在双栏的最后一页上先排满左栏, 然后再排右栏, 可使用命令

```
\raggedend
```

以阻止“平衡”双栏的功能. 恢复平衡双栏的命令是

```
\flushend
```

如果需要插入单栏内容, 只需把单栏内容放在环境

```
\begin{strip}
  单栏内容
\end{strip}
```

之内即可. 参见例 11-1-1.tex.

如果想多栏输出, 或者分栏时不另起一页, 可以使用宏包 `multicol`.

§ 11.2 单、双面

在 `\documentclass` 的可选项中控制单、双面的选项是

```
oneside      twoside
```


可分别称为单面格式选项和双面格式选项. 选用单面格式 `oneside` 时, 所有页码的打印位置是一致的. 选用双面格式 `twoside` 时, 则把奇数页码打印在页面右边. 把偶数页码打印在页面左边.

单、双面选项并不要求实际打印时必须是单面打印或双面打印, 而仅仅是保证当使用了双面选项而又真正在纸张上双面打印时, 装订成册后页码总是在每页的外侧, 翻阅时容易看到.

对于单面格式, 所有页的左边距都相同, 右边距也相同. 对于双面格式, 则是内侧边距都相同, 外侧边距也都相同.

单面格式是 `article` 文档类和 `report` 文档类的默认选项, 双面格式是 `book` 类的默认选项.

对于双面格式, 可将偶数页称为左页, 奇数页称为右页; 对于单面格式, 偶数页与奇数页的页面样式是相同的, 把它们统统称为右页.

对于 `book` 类, 控制每章开始位置的文档类选项是

```
openright    openany
```

默认值是 `openright`, 使每章都是从奇数页开始, 即翻开书时新的一章总是从右边开始, 这有可能产生空白的偶数页. 选项 `openany` 使每章总是从新的一页开始, 而不管这一页是奇数页还是偶数页.

对于双面格式, 隐含使用了内部命令

```
\flushbottom
```

使得每页正文底线处在相同的位置上. 对于单面格式, 隐含使用了内部命令

```
\raggedbottom
```

使得每页正文底线的位置可能稍有上下的变化. 如果在文稿中显式地使用这两个命令, 则可使底线位置与单、双面格式无关.

§ 11.3 与公式有关的选项

在 `\documentclass` 的可选项中, 常用的与公式有关的选项有三个.

行间公式的编号通常都是显示在右边, 如果想把它们统一改到左边, 可使用选项

```
leqno
```

但这个选项只对自动编号起作用, 不改变手工编号的位置.

行间公式通常都是居中对齐, 若都改为左对齐, 可使用选项

```
fleqn
```

当选用了公式左对齐选项 `fleqn` 后, 还可以设置左对齐的位置相对于版心左边界缩进的距离, 命令格式为

```
\setlength{\mathindent}{长度}
```

与设置栏间距的命令类似, 上述命令放在导言区时对整个文档起作用, 放在正文中时则仅有局部作用。

上两个选项对 `$$...$$` 界定的行间公式不起作用, 对 \LaTeX 命令产生的行间公式起作用。

§ 11.4 页版式

11.4.1 页面布局

在一个页面上排放文本的区域分成4部分: 页眉 (Header)、页芯 (Body)、页脚 (Footer) 和边注 (Margin Notes), 在它们的外部是页边空白区域, 各部分的尺寸以及相互之间的距离参见下一页的图形。该图形显示的是在 A4 纸上排版 `article` 类文章时的页面布局, 对于 `book` 类或 `report` 类的页面, 除了各部分的尺寸与 `article` 类有所不同外, 边注区的位置还与奇偶页码有关, 使得装订成书后边注总是在外侧。

控制页面版式的命令通常放在导言区, 命令格式为

```
\pagestyle{版式}
```

其中版式有如下几种:

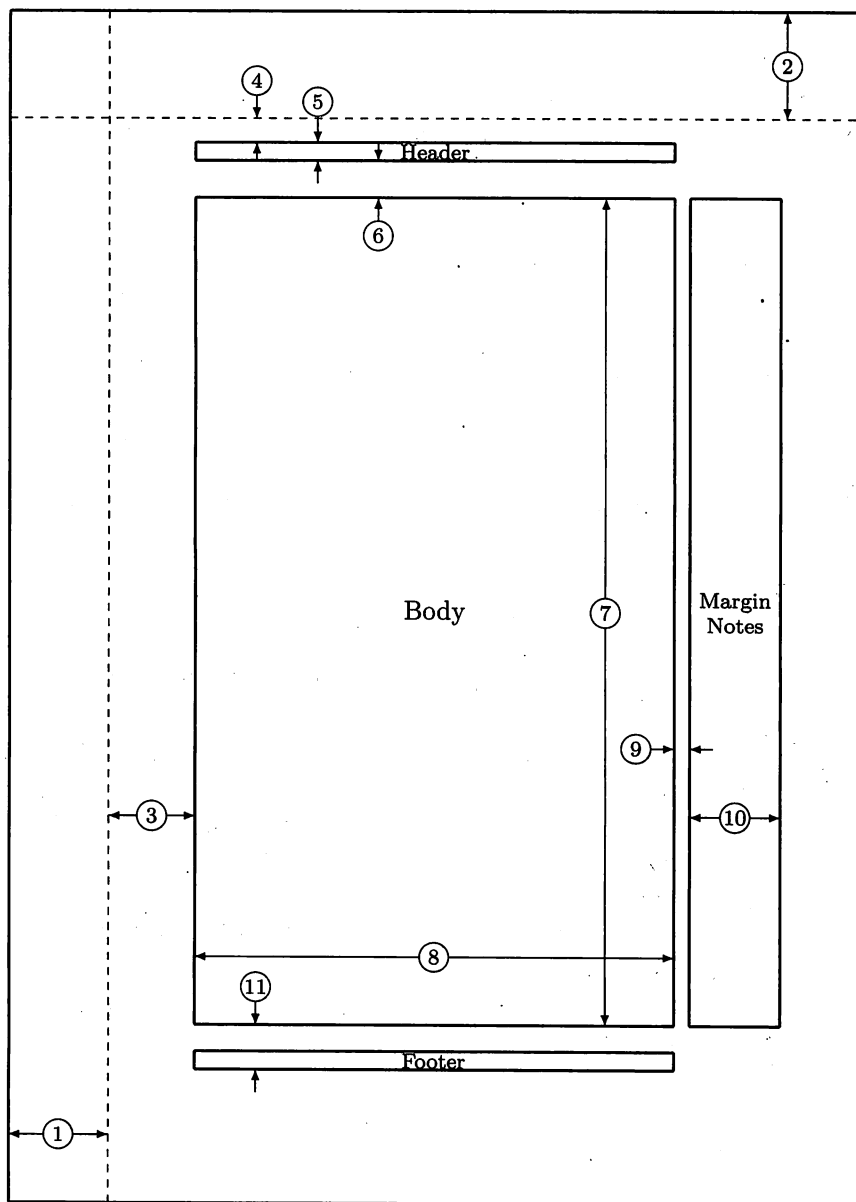
plain 这是默认的页面版式, 即当导言区没有 `\pagestyle` 时, 相当于在导言区放上了命令 `\pagestyle{plain}`。排版输出时页面的页眉区是空的, 页码居中放在页脚区。

empty 页面的页眉和页脚都是空的, 并且不显示页码。

headings 页眉由页码和页眉标题组成, 对于 `book` 类和 `report` 类, 页眉标题含有章和节的标题, 对于 `article` 类, 只有节标题。这些标题是从当前页自动提取的, 但每一章的第一页不显示页眉。

对于不同文档类的双面格式, 左页 (偶数页码) 和右页 (奇数页码) 的页眉内容有所不同, 对于单面格式, 不同文档类的页眉内容也有所不同, 详见下表, 注意单面格式的所有输出页都称为右页。

文档类		左页	右页
book, report	单面格式	——	章
	双面格式	章	节
article	单面格式	——	节
	双面格式	节	小节



- | | | | |
|----|-----------------------|----|----------------------------|
| 1 | one inch + \hoffset | 2 | one inch + \voffset |
| 3 | \oddsidemargin = 62pt | 4 | \topmargin = 16pt |
| 5 | \headheight = 12pt | 6 | \headsep = 25pt |
| 7 | \textheight = 550pt | 8 | \textwidth = 345pt |
| 9 | \marginparsep = 11pt | 10 | \marginparwidth = 65pt |
| 11 | \footskip = 30pt | | \marginparpush = 5pt (未显示) |
| | \hoffset = 0pt | | \voffset = 0pt |
| | \paperwidth = 614pt | | \paperheight = 794pt |

对于 article 文档类, 如果一页上包含的 \section(节) 或 \subsection(小节) 多于一个, 则在左页页眉上显示前一个, 在右页页眉上显示后一个。

myheadings 类似于 headings, 页眉也是由页码和页眉标题组成, 但标题不是自动提取的, 而是由命令 \markright 或 \markboth 决定, 见后面的解释。

如果想要使上述版式仅对当前一页起作用, 则应在当前页的文稿中使用如下命令

```
\thispagestyle{版式}
```

例如 \thispagestyle{empty} 使当前页的页眉和页脚是空的, 不显示页码。注意当前页不显示页码并不影响页码计数器, 下一页的页码与用不用这个命令是无关的。

11.4.2 指定页眉内容

对于页面版式 myheadings, 可用下述命令指定页眉标题内容:

```
\markright{右页页眉}
\markboth{左页页眉}{右页页眉}
```

对于页面版式 headings, 如果不想显示自动提取的页眉, 也可以使用上述命令指定页眉内容。

对于文档类 article 或使用了文档类选项 oneside, 所有的输出页都称为右页 (不考虑页码的奇偶), 因此应使用 \markright 命令。

对于 book 和 report 文档类, 或使用了文档类选项 twoside, 则认为偶数页是左页, 奇数页是右页, 这时应使用 \markboth 命令, 但也可使用命令 \markright 重新定义双面格式时的右页页眉。

对于左页, 页码打印在页眉区的左端, 页眉内容对齐在页眉区右端; 对于右页, 页码打印在页眉区的右端, 页眉内容对齐在页眉区左端。

除非用户直接使用 \markright 或 \markbook 命令指定页眉标题, 否则 L^AT_EX 将使用这两个命令, 提取当前页的章节标题作参数建立页眉标题, 若当前页无新的章节标题, 则继承前面页的章节标题。

\markboth 的第一个参数和它的第二个参数以及 \markright 的参数, 分别存储在下述两个命令中:

```
\leftmark      \rightmark
```

11.4.3 页码

页码的编号可以用阿拉伯数字、罗马数字或拉丁字母, 指定页码样式的命令是

```
\pagenumbering{数字形式}
```

其中数字形式可取如下值 (注意无前导斜线):

arabic	阿拉伯数字 (本页页码就是阿拉伯数字)
roman	小写罗马数字 (如 i, ii, iii, iv 等)
Roman	大写罗马数字 (如 I, II, III, IV 等)
alph	小写拉丁字母 (如 a, b, c, d 等)
Alph	大写拉丁字母 (如 A, B, C, D 等)

默认的数字形式是阿拉伯数字. 需要注意, 每使用一次上述命令, 都会使新页码从 1 开始. 例如在第一章开始放上命令

```
\pagenumbering{roman}
```

在第二章开始放上命令

```
\pagenumbering{arabic}
```

那么第一章的页码是从 i 开始的小写罗马数字, 第二章的页码是从 1 开始的阿拉伯数字.

系统内部有一个页码计数器, 记录当前页的页码. 每生成一页, 计数器就自动加 1, 这个新值就是新页的页码. 如果想使起始页码不从 1 开始, 或者使当前页从新的值开始计数, 可使用如下命令重置计数器的值:

```
\setcounter{page}{页码}
```

其中的页码就是在当前页上要显示的页码.

在 book 类, 命令

```
\frontmatter
```

把页码切换成罗马数字格式, 并且不再对章进行自动编号, 这个命令通常放在前言和目录前面..命令

```
\mainmatter
```

把页码切换成阿拉伯数字, 并对章进行自动编号, 这个命令通常放在正文主体前面. 命令

```
\backmatter
```

通常放在参考文献和索引前面, 它不再对章进行自动编号, 但不改变页码数字格式, 不重设页码计数器.

§ 11.5 自定义页眉和页脚

使用 Piet van Oostrum 编制的 fancyhdr 宏包, 可以容易的设计版面布局, 但本节仅仅介绍其中设计页眉和页脚的命令, 宏包详情可以参看随书光盘的 Doc\常用中文文档\fancyhdr 中文手册 (ifuleyou 译 20010812).pdf. 为使用这个宏包, 应在导言区写上

```
\usepackage{fancyhdr}
\pagestyle{fancy}
```

这个宏包把页面分为奇数页 (Odd page) 和偶数页 (Even page). 把页眉和页脚都分成三个区域, 分别称为左区 (Left field)、中区 (Center field) 和右区 (Right field), 这三个区域之间没有分界线, 放在左区的内容对齐在页眉 (脚) 的左边界, 放在右区的内容对齐在页眉 (脚) 的右边界, 中区的内容则在页眉 (脚) 的整个区域里居中放置, 各区中的内容不可过长, 以避免发生重叠现象.

设置页眉 (header) 和页脚 (footer) 的命令分别是

```
\fancyhead[选项]{内容}
\fancyfoot[选项]{内容}
```

选项是代表区域的字母 L, C, R 与代表奇偶页的字母 O, E 的组合, 例如 LO, RE 等. 如果组合中只有单个字母 O 或 E, 不出现区域字母, 则表示页眉或页脚的三个区域中都要放置同样的内容, 这种情况是罕见的. 如果组合中只有单个字母 L, C 或 R, 不出现奇偶页字母, 则表示无论奇数页还是偶数页, 都在页眉或页脚的指定区域中放置内容, 这种情况是常见的, 为此提供了几个简化命令, 例如 \fancyhead[L] 简化为 \lhead. 类似的有 \chead, \rhead, \lfoot, \cfoot, \rfoot. 此外还有一个更一般的命令 \fancyhf, 它除了可使用上述选项字母外, 还可使用字母 H 或 F, 实际上, \fancyhead 就是带有选项 H 的 \fancyhf, \fancyfoot 就是带有选项 F 的 \fancyhf.

注意命令的次序, 当上下两条命令有冲突时 (例如都对同一区域写内容), 则后出现的命令在冲突处起作用.

命令

```
\headrulewidth \footrulewidth
```

分别表示页眉和页脚与正文之间装饰性分隔线的粗细, 默认值分别是 0.4pt 和 0pt. 改变其值需要使用 \renewcommand 命令而不是 \setlength 命令.

11.5.1 一个简单例子

一篇简单文章, 未分章节, 只想在页眉和页脚的两端分别用中英文显示单位名称, 带有页眉和页脚的装饰分隔线, 并在页脚中央用大写罗马数字显示页码, 形如

East China Normal University		华东师范大学
正文		
Dept. of Math.	I	数学系

可如下设置页眉和页脚(完整文件参见11-5-1.tex):

```

\begin{document}
\begin{CJK}{GBK}{song}

\fancyhf{} % 清空页眉和页脚
\lhead{\bfseries East China Normal University}
\rhead{\CJKfamily{hei} 华东师范大学}
\lfoot{\ttfamily Dept. of Math.}
\rfoot{\CJKfamily{kai} 数学系}
\cfoot{\thepage}
\renewcommand{\headrulewidth}{0.6pt}
\renewcommand{\footrulewidth}{0.4pt}
\pagenumbering{Roman}

正文

\newpage
\end{CJK}
\end{document}

```

11.5.2 使用章节标题作页眉

对于中文书籍,默认的一些章节标题格式不符合中文习惯,全部改成中文格式,要做大量工作,对此有兴趣者请参考随书光盘中的有关资料.如果想做尽量少的改动而又基本符合中文习惯,可增加如下一些命令(需放在正文的CJK环境中):

```
\renewcommand{\chaptername}{}
```

```
\renewcommand{\chaptermark}[1]{\markboth{第\, \thechapter\, 章\ #1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection\ #1}{}}
```

页眉的内容保存在命令 `\leftmark` 和 `\rightmark` 中, 当使用 `book` 文档类时, 它们分别代表当前的章标题和节标题; 对于 `article` 类, 则分别代表当前的节和小节的标题。

配合下述命令

```
\fancyhf{}
\fancyhead[ER]{\leftmark}
\fancyhead[OL]{\rightmark}
\fancyhead[EL,OR]{\$\cdots\ \thepage\ $\cdots\$}
\renewcommand{\headrulewidth}{0.8pt}
```

可以排版出类似本书的页眉 (11-5-2.tex)。

但在正文中, 章的标题被分成两行且为左对齐, 若要变成一行且居中, 可再添加以下命令:

```
\makeatletter
\renewcommand*{\@makechapterhead}[1]{%
  \vspace*{50pt}
  \centerline{\LARGE\bfseries第\, \thechapter\, 章\ #1}
  \vspace{40pt}
}
\makeatother
```

其中 `\@makechapterhead` 是文档类文件 `book.cls` 原有的命令, 用于产生正文中的章标题, 此处把它重定义以符合中文习惯, 因此不能改变这个命令的名字。由于命令中出现了字符 `@`, 而这个命令又被放在了 `TeX` 原稿文件而不是格式文件中, 所以要在它的前后分别加上 `\makeatletter` 和 `\makeatother`, 这是 `LaTeX` 的内部规定, 不要省略。示例文件见 11-5-3.tex。

§ 11.6 目录表及图表清单

在 `LaTeX` 中, 使用命令

```
\tableofcontents
```

可以自动生成包含章节标题和对应页码的目录表, 并把目录表显示在这个命令出现的地方, 该命令通常是放在文章题目之后, 正文之前。

出现在目录表中的章节层次深度可在导言中用如下命令设定:


```
\setcounter{tocdepth}{数}
```

其中的数与第 71 页介绍的计数器 `secnumdepth` 的数含义相同, 默认值也相同.

目录表的内容只有在全部文档处理结束后才能完全确定, 因此含有命令 `\tableofcontents` 的文档常常需要编译两到三次. 第一次是在遇到这个命令时, 创建一个与文稿源文件主名同名但扩展名为 `toc` 的目录文件, 然后在后继处理中把遇到的章节信息加到目录文件中, 第二次编译时读入目录文件并打印在对应位置. 如果 `\tableofcontents` 命令未放在文档开头, 那么第二次编译时还要把它前面的章节信息写到目录文件中, 这就需要第三次的编译才能最后完成任务. 当目录文件已经存在, 文档又未修改时, 只编译一次就够了.

为了把自动编号以外的条目插入到目录表中, 可以使用下列任何一种命令:

```
\addcontentsline{toc}{章节名称}{条目内容}
\addtocontents{toc}{条目内容}
```

第一条命令中的章节名称是去掉前缀倒斜线的章节命令, 用以指定该条目所对应的章节层次. 不同的章节层次在目录表里的默认格式是不同的, 如所用的字体字号是不同的, 缩进的距离也是不同的. 条目内容及其所在页码会按着指定的章节名称的默认格式插入到目录表中.

第二条命令中的条目内容可以含有任何命令和文本, 它们没有任何默认的目录格式, 也不会自动打印该命令所在位置的页码, 当打印目录表时完全是按着条目内容本身包含的命令和文本进行排版.

除了目录表, 插图和表格的列表清单也可由 \LaTeX 自动生成和打印出来. 清单文件的主名与文稿源文件的主名相同, 插图清单文件的扩展名是 `lof`, 表格清单文件的扩展名是 `lot`, 读入或生成这两种文件的两个命令是

```
\listoffigures \listoftables
```

列表清单中的条目是根据 `figure` 或 `table` 环境中 `\caption` 命令的参数自动生成的. 也可以象目录表那样手工插入其他条目, 命令的一般格式是

```
\addcontentsline{清单类型}{格式}{条目内容}
\addtocontents{清单类型}{条目内容}
```

其中的清单类型可以是 `toc` (目录表)、`lof` (插图 清单) 或 `lot` (表格清单). 格式可以是章节名称、`figure` 或 `table`.

为了使各种目录清单的默认标题名为汉字, 应加入下述命令:

```
\renewcommand*{\contentsname}{目\quad 录}
\renewcommand*{\listfigurename}{插图目录}
\renewcommand*{\listtablename}{表格目录}
```

§ 11.7 文档的分割处理

对于一篇小文章, 把它放在一个文件里处理起来很方便, 但若是写一本书, 数百上千页的文字, 如果再放在一个文件里, 处理起来就会很慢很笨拙, 此时应该分成几个文件分别处理, 最后再由 \LaTeX 将结果合并起来.

11.7.1 $\backslash\text{input}$ 命令

使用命令

```
 $\backslash\text{input}\{\text{文件名}\}$ 
```

可以把名为文件名的文件内容读到当前文档的当前位置. 文件名可以带有扩展名, 如果不写扩展名, 则默认扩展名是 `tex`.

$\backslash\text{input}$ 命令可以放在文档的任何部分, 因此可以把整个导言区放到一个单独的文件中, 当很多文件使用同样的导言区命令时, 不必重复输入, 只要用一条 $\backslash\text{input}$ 命令把导言区文件读进来就可以了, 这不仅仅是节约工作量, 更重要的是可以保证各个文件导言区命令的一致性.

一个用 $\backslash\text{input}$ 命令读入的文件中也可以含有 $\backslash\text{input}$ 命令, 嵌套深度只受计算机能力的限制.

为了得到所有读入文件的清单, 可以把命令

```
 $\backslash\text{listfiles}$ 
```

放在导言区. 当处理完毕, 这个清单会同时出现在屏幕上和记录文件(log 文件)中.

11.7.2 $\backslash\text{include}$ 命令

把文档分成几个文件, 对于输入和编辑来说比较方便, 但是当通过 $\backslash\text{input}$ 命令把它们合并起来后, 处理的仍是整个文档, 这样即使在其中一个文件中进行了一点改动, 所有的文件都要被重新读入和处理. 因此人们希望能有一种方法, 可以只重新处理被修改的那个文件.

这可以使用命令

```
 $\backslash\text{include}\{\text{文件名}\}$ 
```

它的作用相当于 $\backslash\text{clearpage}\backslash\text{input}\{\text{文件名}\}\backslash\text{clearpage}$, 只能用于文档的正文部分. 命令 $\backslash\text{clearpage}$ 的作用见第 95 页. 如果只使用这条命令将多个文件包含进来, 则它并不比 $\backslash\text{input}$ 命令好, 反倒有一些局限性, 一是 $\backslash\text{include}$ 命令不可以嵌套: 它只能位于主处理文件中, 但 $\backslash\text{input}$ 命令可以出现在 $\backslash\text{include}$ 进来的文件中. 此外, 它总是开始新的一页然后才读入内容, 不过这并无大碍, 因为人们通常总是在文档应该在开始新页的地方分割文件, 比如章与章之间.

此命令只有与下述命令配合使用才能看出它的优点.

`\includeonly{文件清单}`

这条命令只能位于导言区, 文件清单包含了需要被读入的文件. 文件名之间用逗号分开, 省略扩展名 `tex`.

在文档中可能包含很多的 `\include{文件名}` 命令, 只有文件名出现在文件清单中, 或者在导言区没有 `\includeonly` 命令, 那么 `\include{文件名}` 才真正读入文件, 等价于 `\clearpage\input{文件名}\clearpage`. 如果文件名不包含在文件清单中, 则 `\include{文件名}` 仅相当于 `\clearpage`, 其中内容不被读入. 因此可以通过 `\includeonly` 命令处理选定的几个文件.

此外这条命令保存有关页面、章节和公式编号等信息. 因此选择部分文件处理时来自于其他文件中交叉引用信息也是可用的, 即 `\ref` 和 `\pageref` 命令生成正确的结果. 所有这些值是由前面一次完整处理(用 \LaTeX 编译文件)确定下来的.

对于长文档, `\include` 命令是相当有用的, 它可以节省相当可观的用机时间. 长文档在输入和编辑时通常要分很多步. `\include` 命令可以使得在很短时间内有选择地重新处理改动之处.

如果对被选择处理的文件进行了修改, 导致了页码增加或减少, 或者各种编号有了变化, 则其他的文件也需要被重新处理以校正页码和编号, 这只要随后进行一次完整的处理, 或暂时注释掉导言区的 `\includeonly` 命令就可以做到这一点.

用 `\include` 读入的文件中不能包含任一 `\newcounter` 声明. 这并不是一个过分的限制, 因为通常它们都放在导言区.

本书的每一章就是用单独的文件输入的, 它们名称分别是 `chap1.tex`, `chap2.tex`, ..., 整体处理文件 `all.tex` 本身包含如下文本:

```
\input preamble
\includeonly{.....}
%\typein[\files]{Which files? (for example: chap4, chap5)}
%\includeonly{\files}
\begin{document}
\begin{CJK}{GBK}{song}
\CJKtilde
\input HZcaption
\include{cover}
\frontmatter
\tableofcontents
\mainmatter
\include{chap1}
\include{chap2}
.....
\backmatter
\printindex
```

```

\newpage
\end{CJK}
\end{document}

```

其中 `preamble.tex` 存放所有导言区命令. 只须对 `all.tex` 进行编译. 对于用 `\input` 或 `\include` 引入的每个源文件, 其最后都应写上命令 `\endinput`. 文件 `HZcaption.tex` 的主要内容为

```

\renewcommand*\partname{篇}
\renewcommand*\chaptername{章}
\renewcommand*\appendixname{附录}
%\renewcommand*\refname{参考文献}
\renewcommand*\bibname{参考文献}
\renewcommand*\indexname{索引}
\renewcommand*\figurename{图}
\renewcommand*\tablename{表}
\renewcommand*\contentsname{目录}
\renewcommand*\listfigurename{插图目录}
\renewcommand*\listtablename{表格目录}

```

11.7.3 人机交互命令

在上一节的例子中有一个命令 `\typein`, 这是人机交互命令:

```
\typein[\命令名]{信息}
```

它把信息显示在终端上, 但是它会等待用户从键盘上输入一行文本, 用回车结束. 如果没有可选项 `\命令名`, 那么这文本就直接插入在处理过程中. 如果包含可选项, 那么就认为它等价于 `\typeout{信息}\newcommand{\命令名}{输入的内容}`, 这样就会交互式的把输入的内容保存在名称为 `\命令名` 的命令中, 可以在文档的其他部分同其他 \LaTeX 命令一样调用和执行它. 编译时会在屏幕上显示信息和 `@typein=`, 等待你的输入.

有的时候希望在处理过程中 \LaTeX 能在计算机终端上显示出一条消息, 但不需要用户的输入, 这可以用如下命令:

```
\typeout{信息}
```

这里的信息就代表要显示在计算机屏幕上的文本. 当 \LaTeX 处理到这条命令时, 就会显示出这段文本. 同时, 消息也写入到 `log` 文件中. 如果信息中包含命令 (用户定义的或者 \LaTeX 原有的), 那么命令要被解释, 并把结果显示在屏幕上. 如果命令并不是可显示的, 很可能会造成不良的后果. 要显示命令名而又不执行它, 可在命令的前面加上 `\protect` 命令.

§ 11.8 交叉引用

在长的文本中我们经常需要引用在其他地方出现的章节、表格、插图和公式的编号与页码。对于书籍也需要生成一个索引记录, 它是对整篇文档中特定关键词的引用。在电子文本处理以前的时代中, 如此的交叉引用和索引意味着要耗费作者或其他工作人员大量的精力和时间。现在计算机可以承担这一工作的绝大部分。在以前岁月里, 引用前面文本部分的页码虽然是费事的, 但总是可行的。但是为了说明将要讲述的内容, 引用还没有写出来的部分就只能局限于章节编号了, 因为页码还不知道, 或者留下空白以备将来填上页码。

编写一本书通常是一个渐进的分阶段的工作。手稿可能根本就不是按顺序写的, 而且当初稿完成后, 由于作者新的考虑或者来自于评论者的中肯建议, 有可能对它进行很大的修改、修订、删除或插入某些节、甚至某些章, 交换文本某些部分的顺序更是经常发生的。

L^AT_EX 解决了这些改动造成的困难, 它会自动跟踪所有的改动, 保存交叉引用和索引记录的正确信息, 以供在正文的任何地方引用。

11.8.1 交叉引用

在第 33 页介绍了参考文献的引用方法, 在第 59 页介绍了公式的引用方法, 文本的其他内容也可交叉引用。为此, 需要在被引用的位置设置一个标记, 命令为

```
\label{标记}
```

其中 标记 可以是字母、数字和其他符号的任意组合, 但第 11 页介绍的几个有特殊功能的字符除外。

使用命令

```
\pageref{标记}
```

就会把 标记 所在的页码插入到文本的当前位置。

如果 \label 命令定义在章、节后面或者在公式、枚举罗列 (enumerate) 等环境中, 或者在 figure、table 环境中 caption 的参数值中, 则命令

```
\ref{标记}
```

就会显示出 标记 所在位置相应对象的编号。对于由 \newtheorem 创建的定理型结构 (第 135 页), 如果 \label 命令出现在定理的内容中, 那么定理的编号也可以被引用。例如在某一页上输入了

```
\begin{theorem} \label{th:fermat} ... \end{theorem}
```

在另外的地方输入文本

```
参见第~\pageref{th:fermat}~页定理~\ref{th:fermat}
```

编译后的输出结果就可能形如“参见第 135 页定理 9.1”。

标记命令 `\label` 可以放在章节命令的参数值内或紧接在章节命令之后, 相应的引用命令 `\ref` 显示的编号是该章节的编号. 如果标记命令放在正文中, 相应的引用命令显示的是包含标记命令的最内层的那一节的编号.

L^AT_EX 处理文档时, 遇到标记命令 `\label` 就将其参数标记连同相应的计数器的值和当前页码存放在一个辅助文件中, 该辅助文件的主名与正在处理的文档主名相同, 扩展名是 `aux`. `\ref` 和 `\pageref` 命令就是从这个辅助文件得到有关信息的. 第一次编译时仅是收集信息建立辅助文件, 所以不会输出引用信息而是显示两个问号, 再编译一次就成了.

11.8.2 索引记录

为了在文章或书籍的后面打印索引, 简单的做法需要以下一些步骤. 首先, 在输入文本时, 在导言区写上

```
\usepackage{makeidx}
\makeindex
```

其中命令 `\makeindex` 的作用是激活索引功能, 如果用 `%` 把这个命令注解掉, L^AT_EX 将不理睬文稿中有关索引的命令, 似乎它们根本不存在. 这样可在草稿阶段省去调试改错的时间, 等到最后定稿时再把注解号去掉, 激活索引功能.

在需要打印索引的位置, 一般是在文章的最后, 命令 `\end{document}` 之前, 写上

```
\printindex
```

在被索引的条目位置处写上

```
\index{索引条目}
```

这里的索引条目可以是任何文本, 它将来就是出现在后面索引中的一项. 它可以包含字母、数字和各种符号, 也可以是汉字. 对索引条目可使用各种字体. 用 L^AT_EX 编译后, 就会产生一个主名与所处理文件主名相同但扩展名为 `idx` 的文件.

然后使用 `makeindex` 程序处理生成的 `.idx` 的文件, 即把该文件作为程序的命令行参数:

```
makeindex 文件名
```

可不写扩展名. 运行后产生主名与所处理文件主名相同但扩展名为 `ind` 的文件.

接着再用 L^AT_EX 编译一次主文件就大功告成了, 你会看到类似于本书后每页分成两列打印的索引.

索引页默认的标题是 `Index`, 可以改成中文, 只需使用下述命令

```
\renewcommand{\indexname}{索引}
```

接在命令 `\index` 后面的索引条目可以包含字母、数字和各种符号, 甚至包含 $\text{T}_{\text{E}}\text{X}$ 的命令, 例如: `\index{\section}`, `\index{\[}`, `\index{\%}` 都可以, 但是其中出现的花括号必须配对, 例如 `\index{\{\}}` 是允许的, 而 `\index{\}` 就不可以. 不过 $\text{T}_{\text{E}}\text{X}$ 的命令在排序的时候被忽略, 而且含有汉字的条目在排序时也会有问题, 这时可使用以下形式的命令:

```
\index{排序用条目@打印用条目}
```

例如 `\index{put@\verb=\put=}` 表示此条目按 `put` 被排序, 而打印出来的条目却是 `\put`. 这样就能解决汉字的排序问题, 因为 `Makeindex` 的排序原则是参照 ASCII 的编码, 首先是符号, 然后是数字, 最后是字母. 同一个字母, 大写字母排在小写字母之前 (参见本书末尾的索引, 注意我们在排序用条目栏里除去了 ‘\’ 号). 为了使汉字按汉语拼音顺序排在英文字母之后, 我们可用以下的办法输入

```
\index{zzzzsuoyin@索引}
```

还要注意字符 ‘!’, ‘@’, ‘|’ 在 `MakeIndex` 里是有特殊功能的命令字符, 因此不能按它们的本来意义出现在索引条目内, 详情可参见 `\texmf\doc\makeindex` 目录内的说明文件.

第十二章 图形包介绍

在第六章介绍了 \LaTeX 的绘图命令,可以用来绘制直线、矢量线、圆、矩形、圆角矩形(卵形线)、Bézier曲线等基本图形.使用天元绘图软件,还可画出参数方程的二维或三维曲线的图形.除此而外还有很多图形软件包用于绘制或者插入图形.

本章除了介绍基本图形包`graphics`的用法外,还介绍一些有用的图形包.对于需要画交换图的读者来说,`amscd`最容易使用,但是功能也最差,基本上只能画方图,没有斜向的箭头.`Diagrams`和`Xy-pic`的功能都相当强,容易上手.尤其`Xy-pic`是AMS认可的,与AMS的宏包配合使用,可以用于向国外投稿.

如果你需要画复杂的图形,那么`MetaPost`、`PSTricks`和`pgf`都是合适的选择.其中`MetaPost`是仿照`METAFONT`的风格,它会解线性方程组求出两条直线的交点,确定两点的定比分点,还能做仿射变换,功能相当强大.而且能与`pdfLaTeX`配合,生成pdf文件.缺点之一是入门不易,因为它实际上是一个高级编程语言,一旦熟练了,就能画出复杂的图形.另一个缺点是`MetaPost`的图形必须放在另一个单独的文件里,不便维护和封装.而且`MetaPost`基本上已经不再发展.

`PSTricks`的功能十分强大,而且处于不断发展之中,不同作者开发的同族宏包越来越多,可以根据不同的需要选用,因此前景看好.经常需要绘制复杂图形的读者不妨试试.它的缺点是与`pdfLaTeX`不兼容,要使用`pdfticks`包作转换.它与演示用的宏包`beamer`也不兼容.此外如果要画函数曲线时,函数方程的解析式要用PostScript的命令格式输入,与我们通常输入公式的习惯完全不同.

`Pgf`是一个较新的、仍在发展中的宏包,它是`beamer`的作者开发的,因此两者完全兼容.它的特点是兼容ps与pdf,既能用于dvips,又能用于`pdfLaTeX`.缺点是功能尚不及`PSTricks`强大,例如没有由方程画曲线的功能.

以上图形包都采用坐标定位,因此用户预先需要有一个总体设计,而且不是所见即所得的.对于懒于计算的读者缺乏吸引力.这类读者不妨试试`WinFIG`,它是`Xfig`的Windows翻版.我们收录了一个版本,读者可在随书光盘的`Extras\WinFIG1.5`找到.`WinFIG`是所见即所得的,与 \LaTeX 有很好的配合.其缺点是不能标注数学公式,精确性也差,这是所见即所得软件的共同缺点.请读者自己摸索其安装与使用方法,本书不再介绍.

§ 12.1 图形包 graphics

在 L^AT_EX 中已开发出了几个图形宏包, 如“标准”的 graphics 宏包和“扩展”的 graphicx 宏包. 常用的还有 color 宏包. 利用这些宏包可以容易地插入外部彩色或黑白图形, 并可进行裁剪、放缩、翻转和旋转. 在 graphics 和 graphicx 中, 定义的命令和命令的不可省略参数都是相同的, 差别仅在于可选项的格式和内容. 两个宏包为所有的驱动程序提供了一组通用的命令, 而与驱动程序有关的代码存放在特定的 def 文件中, 通过宏包的选项加载它们, 因此只要改变选项, 就可以切换到另一种驱动程序, 正文本身无需修改. 因为标准的 graphics 宏包更符合 L^AT_EX 的语法, 所以介绍这个宏包, 关于宏包 graphicx 的介绍可参见第 266 页. 使用宏包时, 应先在导言区写上

```
\usepackage[选项]{graphics}
\usepackage[选项]{color}
```

其中选项处应包含驱动程序名, 可用的有

dvipdf	dviwindo	pctexhp	tcidvi
dvips	emtex	pctexps	textures
dvipsone	oztex	pctexwin	truotex
dviwin	pctex32	pdftex	xdvi

其中 dviwin, emtex, pctexhp, pctexwin 不支持彩色、放缩和旋转, truotex 不支持放缩和旋转. 在 Linux 系统中应使用选项 xdvi, 在 Windows 系统中多数情况下可使用 dvips 选项, 当将 dvi 文件转换成 ps 文件后, 可使得插入的外部图形可在支持 ps 文件的设备上显示或打印出来. 如果使用 pdfL^AT_EX (见 § 13.2) 直接由 tex 文件生成 pdf 文件, 则应使用选项 pdftex. WinEdt 集成环境含有工具按钮可很方便地将 dvi 文件转换成 ps 文件, 或直接生成 pdf 文件. 为了明确起见, 除非特别说明, 本节总是假设在调用图形宏包时使用了 dvips 选项.

除了驱动程序名, 用于 graphics 宏包的其他的一些选项有

draft 不插入图形, 仅保留出图形所占位置, 显示图形边界方框及图形文件名. 这可明显加快处理速度.

final 真正插入图形. 这是默认值. 但若在 \documentclass 选项中使用了 draft, 则必须显式写上选项 final 才能真正插入图形.

hidescale 当对图形放缩时, 只保留放缩后所占区域而不插入图形. 当预览器不支持放缩时必须使用该选项.

hiderotate 当对图形旋转时, 只保留旋转后所占区域而不插入图形. 当预览器不支持旋转时必须使用该选项.

hiresbb 在插入的 eps 图形文件中不使用通常的 %BoundingBox 的值, 而是用 %HiresBoundingBox 的值作为图边界盒子的值. 边界盒子 (BoundingBox) 是包含图形的正矩形区域, 正矩形是四边横平竖直的矩形, 不是倾斜的矩形.

monochrome 用于 color 宏包的选项, 使彩色命令不起作用. 当预览器不支持彩色时应使用该选项.

当两个宏包的选项完全相同时, 例如都只有一个驱动程序选项, 可将它们放在同一个 \usepackage 命令里.

图形宏包支持最好的图形格式是 EPS (Encapsulated PostScript) 格式, 本节就以这种格式的图形文件作例子. 将其他格式的图形文件转换成 EPS 格式是很容易的, 很多软件都可进行图形格式转换, 例如功能很强的免费软件 ImageMagick, 可从 www.imagemagick.org 或 ftp.wizards.dupont.com 等站点自由下载.

EPS 图形文件的头部形如

```
%!PS-Adobe-3.0 EPSF-3.0
%%Creator: Adobe Photoshop Version 6.0
%%Title: panda2.eps
%%CreationDate: Sat Oct 27 2001 17:16:27
%%BoundingBox: 0 0 59 59
%%HiResBoundingBox: 0 0 59.25 59.25
%%SuppressDotGainCompensation
%%DocumentProcessColors: Black
%%EndComments
```

其中边界盒子 (BoundingBox) 的数据反映了图形的大小. 上例中的数据是 0 0 59 59, 表示边界盒子左下角坐标为 (0, 0), 右上角坐标为 (59, 59), 单位是“大点” bp (big point), $1\text{in}=72\text{bp}=72.27\text{pt}$. 容易算出这个图形的宽与高均为 59bp, 约为 0.819 英寸, 即 2.08 厘米.

图形盒子左下角坐标不一定是 (0, 0), 例如文件 pic.eps 的头部几行如下:

```
%!PS-Adobe-2.0 EPSF-1.2
%%Creator: MATLAB, The MathWorks, Inc
%%Title: MATLAB graph
%%CreationDate: 03/09/98 17:26:21
%%Pages: 001
%%BoundingBox: 080 407 532 756
%%DocumentFonts: Times-Roman
%%DocumentNeededFonts: Times-Roman
%%EndComments
```

边界盒子左下角坐标为 (80, 407), 右上角坐标是 (532, 756), 图形宽度应是 $532 - 080 = 452\text{ bp}$, 高度是 $756 - 407 = 349\text{ bp}$. 可以想像有一个坐标系, 边界盒子的坐标以及下文提到的剪裁坐标都是相对于这个坐标系的. 这个坐标系的原点实际是纸张的左下角.

12.1.1 插入图形的基本命令

插入图形的基本命令是

```
\includegraphics*[左下角x,y][右上角x,y]{图形文件}
\includegraphics[左下角x,y][右上角x,y]{图形文件}
```

其中带 * 号的命令实际插入剪裁后的图形, 剪裁区域的左下角与右上角的坐标由命令中的可选项决定. 如果省略了左下角坐标, 则默认左下角坐标取 (0, 0). 如果不写可选项, 则通过读取边界盒子的值确定图形大小. 坐标可以带单位, 无单位时默认单位是 bp. 上述坐标仅仅是用于确定剪裁区域的大小及在原来图形上的位置, 当将其插入到文档时, 是插在当前位置, 与原来的坐标系没有关系.

不带 * 号的命令是插入整个图形, 剪裁范围的坐标仅仅用于确定图形占用的区域, 当剪裁范围与图形边界盒子不相符时, 有可能使插入的图形与周边文字重叠或出现过多的空白.

有些驱动程序不支持图形的剪裁, 或者不支持某些格式图形的剪裁, 此时上述两个命令有可能产生意外的效果, 比如剪裁不起作用, 总是插入整个图形, 或者将整个图形放缩到剪裁区域的大小. 后面介绍的其他命令也会有类似问题, 不再一一赘述.

下面是一个例子 (参见 12-1-1.tex), 特意加了方框以看出剪裁区域的范围:

```
\setlength{\fboxsep}{0pt}
\begin{center}
\fbbox{\includegraphics*[10,10][30,40]{pic/panda.eps}}\qqquad
\fbbox{\includegraphics[10,10][30,40]{pic/panda.eps}}\qqquad
\fbbox{\includegraphics{pic/panda.eps}}
\end{center}}
```

排版输出结果是



可以看出, 对齐位置和相互间距都是取决于裁剪区域的.

12.1.2 放大和缩小

按比例进行放缩的命令是

```
\scalebox{横向放缩因子}[竖向放缩因子]{插入的图形}
```

其中竖向放缩因子是可选项, 若不用该项, 则竖向与横向用相同的放缩因子, 此时可保持图形横竖比例不变. 放缩因子可以是负数, 表示向反方向放缩. 插入的图形既可以是整个图形, 也可以是剪裁后的图形. 输入 (参见 12-1-2.tex)

```
\begin{center}
\scalebox{0.4}[0.2]{\includegraphics{pic/pic.eps}}\quad
\scalebox{0.2}{\includegraphics{pic/pic.eps}}
\end{center}
```

输出



使用这个命令可以显示 扁体字 和 瘦体字, 此处使用的命令为

```
\scalebox{2.0}[0.5]{扁体字} 和 \scalebox{0.5}[2.0]{瘦体字}
```

放缩到指定大小的命令是

```
\resizebox{宽度}{高度}{插入的图形}
```

其中的 {宽度} 和 {高度} 都要指定, 但其中一个可以是 {!}, 这表示在一个方向上放缩到指定尺寸时, 仍保持图形原有的横竖比例不变. 输入 (参见 12-1-3.tex)

```
\begin{center}
\resizebox{4cm}{25mm}{\includegraphics{pic/pic.eps}}\quad
\resizebox{1.2in}{!}{\includegraphics*[100,430][400,600]{pic/pic.eps}}
\end{center}
```

输出为



12.1.3 水平翻转和旋转

将图形水平翻转的命令是

```
\reflectbox{插入的图形}
```

这实际上是命令 `\scalebox{-1}[1]{..}` 的缩写. 输入 (参见 12-1-4.tex)

```
\begin{center}
\reflectbox{\includegraphics{pic/panda.eps}}\quad
\reflectbox{\includegraphics*[10,10][30,40]{pic/panda.eps}}
\end{center}
```

输出为 (注意原来头朝左, 现在头朝右)



用这个命令可以得到镜像字, 例如:

```
\newsavebox{\tmpbox}
\sbox{\tmpbox}{\Huge\HEI\shortstack{镜\\像}}
\usebox{\tmpbox}\rule{5pt}{45pt}%
\reflectbox{\usebox{\tmpbox}}
```

镜
像

将图形旋转的命令是

```
\rotatebox{旋转角度}{插入的图形}
```

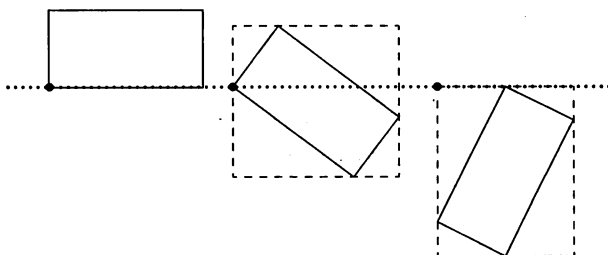
其中旋转角度以度为单位, 逆时针方向为正. 例如输入 (参见 12-1-5.tex)

```
\begin{center}
\includegraphics{pic/panda.eps}
\rotatebox{-30}{\includegraphics{pic/panda.eps}}
\rotatebox{30}{\includegraphics{pic/panda.eps}}
\rotatebox{180}{\reflectbox{\includegraphics{pic/panda.eps}}}
\end{center}
```

输出为



EPS 图形的盒子是其边界盒子, 基准点在左下角. 图形的旋转就是盒子的旋转, 默认值是绕基准点旋转. 旋转后图形区域的外切正矩形是旋转后的图形盒子, 旋转后图形盒子的高度、深度和宽度都会发生变化. 下图是一个盒子顺时针旋转 37° 和 120° 后的情况, 图中圆点是旋转后的基准点, 基准点所在水平线是基线 (图中用虚线表示), 虚线矩形是旋转后的图形盒子.

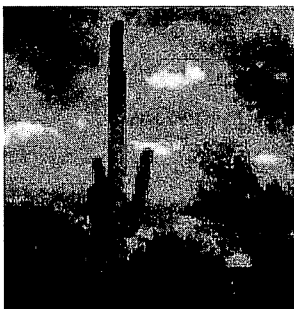


外部图形也可以放在绘图环境中, 例如输入 (参见 12-1-6.tex)

```
\begin{center}
\begin{picture}(75,42)
\put(0,30){这是一幅}
\put(18,0){\resizebox{!}{4cm}{\includegraphics{photo.eps}}}
\put(60,10){风景照片}
\end{picture}
\end{center}
```

输出为

这是一幅

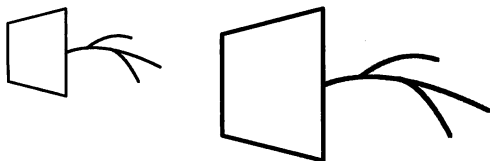


风景照片

上述命令中的参数{插入的图形}都可以换成普通文本或其他对象,例如输入 `\rotatebox{-15}{Graphics}` 显示为 *Graphics* (若在预览器上看不到这种效果,用 `dvips.exe` 转换成 ps 文件后再看).

上述的放缩、旋转等命令可以嵌套使用,但命令的次序对最后结果是有影响的.下面给出一个例子(参见 12-1-7.tex),其中的对象是第 86 页画出一盆花.左边是先顺时针旋转 90 度,再放缩到宽度 2 cm,右边是先放缩到宽度 2 cm,再顺时针旋转 90 度:

```
\begin{center}
\resizebox{2cm}{!}{\rotatebox{-90}{\usebox{\flower}}}
\quad
\rotatebox{-90}{\resizebox{2cm}{!}{\usebox{\flower}}}
\end{center}
```



对显示结果不必惊讶,看一下这盆花的尺寸就好理解了.这个图形原始尺寸是宽 12 pt,高 21 pt,高度几乎是宽度的一倍.左边是先旋转 90 度,旋转后宽度变成了高度的一倍,然后将旋转后的图形的宽度变为 2 cm,高度就只有 1 cm 多点;右边图形是先将宽度变为 2 cm,高度就几乎为 4 cm 了,然后再旋转 90 度,结果的宽度就差不多是 4 cm,比左边的图形宽了近一倍.

§ 12.2 用宏包 amscd 画交换图

我们这里介绍几种画交换图的方法,一种比较简单,使用 \LaTeX 的宏包 `amscd`,可以画出各种方形的交换图.另外的选择是利用宏包 `diagrams` 或 `Xy-pic`,它们可以画出相当复杂的交换图.

为调入宏包 `amscd`,必须在导言中加入以下语句:

```
\usepackage{amscd}
```

我们先观察下面的交换图:

$$\begin{array}{ccc}
 A \amalg B & \xrightarrow{p_A} & A \\
 p_B \downarrow & & \uparrow f \\
 B & \xleftarrow{g} & Z
 \end{array}$$

其输入为

```
\[ \begin{CD}
A\prod B @>p_A>> A \\
@Vp_BVV @AAfA \\
B @<<g< Z
\end{CD} \]
```

可见 @>>> @<<< @AAA @VVV 分别表示向左、向右、向上、向下的箭头, 而出现在第一个 < 或 > 后面的数学符号会用 \scriptstyle 字体打印在水平箭头的上面, 出现在第二个 < 或 > 后面的数学符号则打印在水平箭头的下面。

类似地, 出现在第一个 A 或 V 后面的数学符号会用 \scriptstyle 字体打印在竖直箭头的左边, 出现在第二个 A 或 V 后面的数学符号则打印在竖直箭头的右边。

我们再看下面两个例子:

$$\begin{array}{ccc}
 G & \xlongequal{\quad} & G' \\
 & \parallel & \\
 & H &
 \end{array}$$

其输入为

```
\[ \begin{CD}
G @= G' \\
@. @| \\
@. H
\end{CD} \]
```

这个例子说明命令 @= 与 @| 能分别生成水平或竖直的两条平行线. 此外, 如果位于四角的某个对象不出现, 只要输入 {} 或留空即可, 但是如果某个箭头不出现, 则要输入 "@.", 不能只是留空.

$$\begin{array}{ccc}
 G & \xrightarrow{\text{Clifford multiplication}} & H \\
 f \downarrow & & \uparrow g \\
 G' & \xleftarrow{\beta} & H'
 \end{array}$$

其输入为


```

\[\begin{CD}
G @>\text{Clifford multiplication}>> H \\
@VfVV @AAgA \\
G' @<\phantom{Clifford multiplication}<<\beta< H'
\end{CD} \]

```

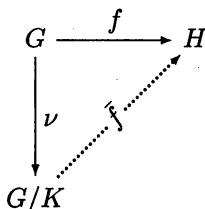
这个例子让我们看到如何利用 T_EX 的占位命令 `\phantom` 使得上下两个箭头有相同的宽度. 这里的 `\text` 命令只能在调入 `amsmath` 宏包后才能使用, 否则要用 `\mbox{\scriptsize ...}` 代替 (参见第 142 页). 我们把本小节使用的例子的原文都收集在 `12-2-1.tex` 里.

§ 12.3 用宏包 diagrams 画交换图

P. Taylor 开发的宏包 `diagrams` 功能十分强大, 要了解它的全部功能, 可以参看文件 `texmf/doc/generic/taylor/manual.pdf`. 这里只介绍一些最容易掌握的命令. 我们把本小节使用的例子的原文都收集在 `12-3-1.tex` 里. 由于随书光盘已经安装了这个宏包, 为了使用 `diagrams`, 只要导言中加入以下语句:

```
\usepackage{diagrams}
```

现在我们先看一个简单的例子:



其输入为

```

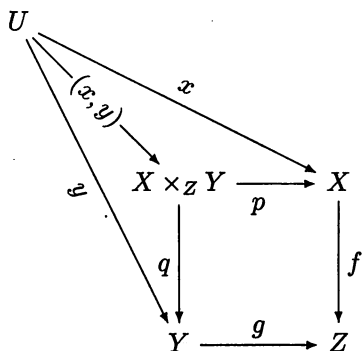
\begin{diagram}
G      & \rTo~f      & H \\
\downarrow \nu & \ruDotsto~{\bar f} & \\
G/K    & & 
\end{diagram}

```

从上面的例子可以看出, `diagram` 环境是把一个交换图看成一个表格, 然后再把对象与箭头分别填入相应的格子中. 与 `Xy-pic` 不同的是, 对象和箭头各占一格, 不能放在同一个格子里. 箭头后面的上标表示注在上面或左面的字符, 下标则表示注在下面或右面的字符, 而注在箭头中间的字符则输入在 `~` 号的后面.

箭头方向的表示方法如下: `\lTo`, `\rTo`, `\uTo`, `\dTo`, `\lTo`, `\ldTo`, `\ruTo`, `\rdTo` 分别表示指向左方、右方、上方、下方、左上方、左下方、右上方、右下方

的箭头, 命令 `\rdTo(4,2)` 表示指向右下方的箭头, 目标位于右向第4格、下方第2格. 请读者参看下面的例子:



相应的输入为

```
\begin{diagram}
U\\
&\rdTo~{(x,y)}\rdTo(4,2)^x\rdTo(2,4)_y\\
&\qquad\qquad\qquad&X\times_Z Y&\rdTo_p&X\\
&\qquad\qquad\qquad&\rdTo_q&\qquad&\rdTo_f\\
&\qquad\qquad\qquad&Y&\rdTo_g&Z
\end{diagram}
```

下表显示了可以使用的各种箭头式样:

<code>\rTo</code>	\longrightarrow	<code>\rLine</code>	$\rule{1cm}{0.4pt}$
<code>\rEmbed</code>	\twoheadrightarrow	<code>\rOnto</code>	\twoheadrightarrow
<code>\rDotsto</code>	$\cdots\rightarrow$	<code>\rDashto</code>	$-\cdots\rightarrow$
<code>\rImplies</code>	\implies	<code>\rMapsto</code>	\mapsto
<code>\rInto</code>	\hookrightarrow	<code>\rProject</code>	\twoheadrightarrow
<code>\hDots</code>	\cdots	<code>\hDashes</code>	$---$
<code>\hEq</code>	\equiv		

表中前5种箭头 `\rTo`, `\rLine`, `\rEmbed`, `\rOnto`, `\rDotsto` 可以使用于其他方向, 包括倾斜的方向; 最后3种线条 `\hDots`, `\hDashes`, `\hEq` 只要把命令中的 `h` 改成 `v`, 就能产生竖直方向的线条; 而其余几种箭头 `\rDashto`, `\rImplies`, `\rMapsto`, `\rInto`, `\rProject` 则只能用于水平与竖直的4个方向.

箭头类型的定义: 除了上面列出的箭头外, 用户可以利用一下命令自己定义箭头.

```
\newarrow{箭头名称}{箭尾}{填充类型}{中段}{填充类型}{箭头}
```

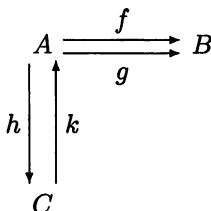
例如:

`\newarrow{EEmbedd}{>>}---{>>}` 可以定义箭头 \Rightarrow ;
`\newarrow{Corresponds}<--->` 可以定义箭头 \longleftrightarrow ;
`\newarrow{Openinto}{triangle}--->` 可以定义箭头 \triangleright ;
`\newarrow{Congruent}33333` 可以定义箭头 \equiv ;
`\newarrow{Backwards}<----` 可以定义箭头 \longleftarrow ;
`\newarrow{Partial}----{harpoon}` 可以定义箭头 \rightharpoonup .

用上述命令定义的箭头在使用时前面应该加上字母 `l`, `r`, `u`, `d`, `lu`, `ld`, `ru`, `rd` 等以表示方向. 例如 `\rEEmbedd`, `\dCongruent` 等.

平行箭头: 竖直方向的平行箭头很容易画, 只要把它们写在同一个格子里就可以了. 而要画水平方向的箭头, 则要利用命令 `\pile`, 如下例所示:

```
\begin{diagram}
  A      & \pile{\rTo^f\ \rTo_g} & B \\\
\downarrow h & \uparrow k & \\
C\end{diagram}
```



设定 `diagram` 环境的可选参数: 有以下 4 种方式给出 `diagram` 环境的可选参数, 它们的作用范围是显然的.

```
\usepackage[可选参数]{diagrams}
\diagramstyle[可选参数]
\begin{diagrams}[可选参数]
\rTo[可选参数]
```

下面列出一些常用的参数, 可以同时选择多个参数, 相互以逗号间隔.

`abut` 在箭头或箭尾与对象之间不留空隙.

`alignlabels` 使水平箭头的标注位于格子的中间, 因此不一定位于箭头的中点. 这是默认的选项.

`amstex` 使得 `amscd` 的交换图命令 (参见 § 12.2) 也能使用.

`balance` 或 `hmiddle` 以图形的中间网格作为水平中线, 不考虑图形的真实水平长度.

`bottom` 以图形最底层的行作为水平对齐的基线.

`centre` 或 `center` 使图形按真实大小居中, 相当于 `nobalance` 与 `vcentre`.

`centredisplay` 或 `centerdisplay` 使图形按 `balance` 的规则作为行间公式居中, 与 `leftflush` 的效果相反.

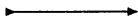
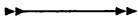

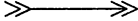

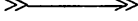
















`dotted` 以点线作为箭身 (仅适用于箭头的命令, 如 `\rTo[dotted]`).

`eqno`=标号 给图形加上标号 (数学模式).

`h`=长度 或 `height`=长度 规定相邻两行间的距离.

`hcentre` 或 `hcenter` 以真实的中心线作为水平方向的中线, 与 `balance` 相反.

`heads`=类型名称 为命令 `\newarrow` 中的 `>` 和 `{>>}` 规定类型, 目前可用的类型有

LaTeX			(默认值)
vee			
littlevee			
triangle			
o			
O			
X			
+			
curlyvee			(使用 AMS 符号)
blacktriangle			(使用 AMS 符号)
littleblack			(使用 AMS 符号)

`inline` 如果选取 `leftflush` 或 `centredisplay` 作为整体参数, 又想把当前图形与其他图形排列在同一行, 则可在这些图形环境里使用此选项.

`l>`=长度 为水平或斜向箭头规定最小长度, 默认值是 `2em`.

`labelstyle`=命令 设定箭头上标注的字形大小, 除了默认值 (`\textstyle`) 之外, 最常用的是 `\scriptstyle`.

`landscape` 在设定了 `PostScript` 选项的前提下, 使得图形旋转 90° .

`large` 相当于 `size=5em`.

`LaTeXeqno` 利用 \LaTeX 的 `equation` 环境的编号系统作为 `eqno`. 在图形环境内部可以使用 \LaTeX 的 `\label` 命令.

`lefteqno` 把编号打印在图形的左端.

`leftflush`[=长度] 使图形居左对齐. 如果注明了长度, 则最左边的竖直箭头到页边的距离等于此长度.

`leftshortfall`=长度 规定箭头左端与左边对象间的间隔.

`lowershortfall`=长度 同上, 不过用于下方.

`loose` 在需要时允许图形扩展.

`middle` 水平方向用 `balance` 方式居中, 竖直方向用 `vmiddle` 方式居中.

`midshaft` 或 `midshaft` 把标注放在水平箭头的箭身中部.

`nobalance` 按图形的两端确定水平方向的中点, 与 `balance` 相反.

`noorigin` 使 `origin` 和 `balance` 失效.

`objectstyle`=命令 设定对象的字形大小, 除默认值外, 最常用的是 `\scriptstyle`.

`origin` 把整个图形的长、宽、深都设为0, 把它放在底部行的基线上, 以最左边的列作为中心.

`p=长度` 或 `pilespace=长度` 设定 `\pile` 命令里水平平行箭头间的距离.

`portrait` 使选项 `landscape` 失效.

`PostScript=dvips` 激活 PostScript.

`righteqno` 把编号放在图形的右侧.

`rightshortfall=长度` 规定箭头右端与右左边对象间的间隔.

`s=长度` 或 `size=长度` 使得 `height` 和 `width` 都等于这个长度.

`scriptlabels` 使箭头的标注都采用 `\scriptstyle`.

`shortfall=长度` 使得箭头的端点与旁边的对象之间保持此间隔.

`small` 相当于 `size=2em`.

`thick[=尺度]` 使得水平或竖直的箭头具有指定的厚度, 默认值相当于 L^AT_EX 的 `picture` 环境里规定的厚度. 但对斜向的箭头不起作用.

`thin[=尺度]` 见上条.

`tight` 使每个网格严格遵守 `size` 指定的尺度. 这不是默认的选项, 但建议在打印最后定稿时使用此选项.

`top` 以图形最顶层的行作为水平对齐的基线.

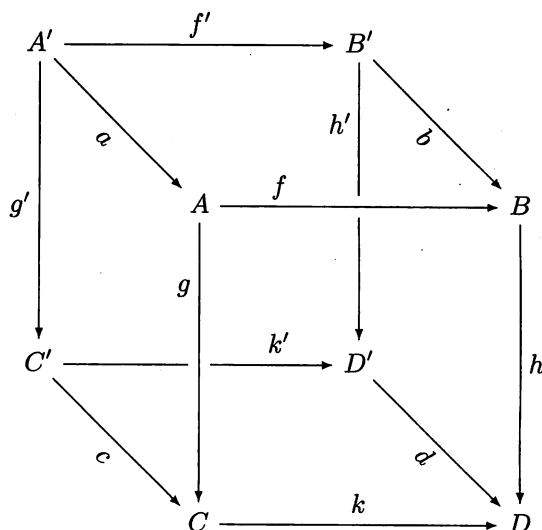
`uppershortfall=长度` 规定箭头顶端与上边对象间的间隔.

`vcentre` 或 `vcenter` 其竖直位置相当于 T_EX 的 `\vcenter`, 或 L^AT_EX 的 `[c]` 选项, 即图形竖直方向的真实中线.

`vmiddle` 当行数是奇数时, 以中间行的基线作为图形的基线.

`w=长度` 或 `width=长度` 设定网格的宽度.

最后再看一个例子:



其输入如下, 注意其中的新命令 `\HonV` 以及 `\VonH` 的意义.

```

\begin{diagram}
  A' & & & \rTo^{f'} & & B' & & \\
      & & \rdTo.a & & & \vLine^{h'} & \rdTo.b \\
\rdTo^{g'} & & A & \rTo^f & \HonV & & B \\
      & & \rdTo.g & & \rdTo & & \\
  C' & \hLine & \VonH & \rTo^{k'} & D' & & \rdTo.h \\
      & \rdTo.c & & & & \rdTo.d \\
      & & C & & \rTo^k & & D
\end{diagram}

```

§ 12.4 Xy-pic

Xy-pic 是 Kristoffer H. Rose 和 Ross Moore 开发的用于绘图, 尤其是交换图的 $\text{T}_\text{E}\text{X}$ 宏包, 是一个自由软件。

由于 Xy-pic 用 METAFONT 建立自己的字模, 因此它作出的图形不依赖 PostScript. Xy-pic 除了在画交换图, 尤其是含有弧线的交换图方面有它的特色外, 它可以画图论中用到的各种有向或无向图, 以及画扭结图, 这些都是它的特色。

使用 Xy-pic 宏包应在导言区写上

```
\usepackage[all]{xy}
```

其中 [all] 表示使用该宏包的常用选项, 此外若画扭结图, 需用 knot 选项, 画有向图需用 poly 选项. 我们把本小节使用的例子的原文都收集在 12-4-1.tex 里, 本节的内容和一些例子取自 Kristoffer H. Rose 的文章 “Xy-pic User’s Guide”.

下面介绍具有矩阵特色的图形, 生成矩阵特色图形的命令是

```
\xymatrix{...}
```

其中 “...” 是横竖对齐的矩阵元素以及元素之间的各类连线, 元素行用 \\ 分隔, 元素列用 & 分隔. 例如不带连线的矩阵)

```

\xymatrix{
  A & *{\text{B}} & C & \\
  {\bullet} & {\sum_{k=1}^n k^3} & {\pi} & \\
}

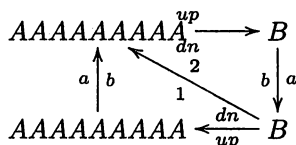
```

的输出结果是

$$\begin{array}{ccc}
 A & B & C \\
 \bullet & \sum_{k=1}^n k^3 & \pi
 \end{array}$$

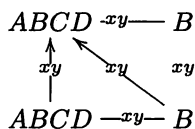
个元素的中心的中间位置), 如果想放在连线(不包括元素)的中部, 需在标记的前面加一个减号(形象地表示“连线”).

```
\xymatrix{
  AAAAAAAA \ar[r]^{up}_{dn} & B \ar[d]^{a_b} \\
  AAAAAAAA \ar[u]^{a_b} & B \ar[l]^{-up}_{dn} \ar[ul]^{1.2}
}
```



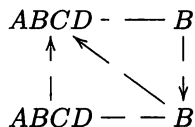
在连线内也可添加标记, 即把连线从中间打断, 断开处写上标记, 此时既不用 \wedge 也不用 \perp , 而是使用竖线 $|$, 注意, 如果想放在连线的中部, 仍需在标记的前面加一个减号. (下面例子有一个 $\@{\}$ 不可见连线)

```
\xymatrix{
  ABCD \ar@{-}[r]|\{xy\} & B \ar@{\hspace{0.5cm}}[d]|\{xy\} \\
  ABCD \ar[u]|\{xy\} & B \ar@{-}[l]|\{xy\} \ar[ul]|\{xy\}
}
```



如果只要连线断开, 出现一个洞, 只需将添加的标记写成 $\backslash hole$, 如果想在连线中部断开, 仍需在标记的前面加一个减号.

```
\xymatrix{
  ABCD \ar@{-}[r]|\backslash hole & B \ar[d]|\backslash hole \\
  ABCD \ar[u]|\backslash hole & B \ar@{-}[l]|\backslash hole \ar[ul]|\backslash hole
}
```



试试下面的图形:

$$\begin{array}{ccc} A & \longrightarrow & B \\ \uparrow & & \downarrow \\ B & \sim & C \end{array} =$$

输入为

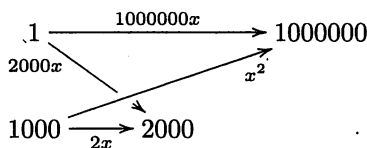
$$\begin{matrix} A \\ B \end{matrix} \begin{matrix} r \\ u \end{matrix} @ \begin{matrix} [d] \\ [r] \end{matrix} = & B \begin{matrix} d \\ u \end{matrix} \backslash \sim C$$

明确指定标记位置的符号除了表示连线中点的单个-号外, 还可由单个或多个的<或>号。单个<号表示把标记放在向量的起点, 每多一个<号, 标记的位置就向终点移动一个\jot (约3pt) 距离; 单个>号表示把标记放在向量的终点, 每多一个>号, 标记的位置就向起点移动一个\jot 距离。

另一种指定位置的方法是用一个放在圆括号中的小数表示, 起点元素的中点是 (0), 终点元素的中点是 (1), 如果使用了单个或更多的 < 或 > 号, 则 (0) 或 (1) 指的是向量上由这些小于号或大于号确定的位置. 表示连线中点位置的减号相当于 <>(0.5).

还有一种指定直的(不弯曲的)向量上的位置的方法: $! \{p; q\}$, 其中 p, q 表示两个元素(用绝对标号或相对标号), 指定的位置就是通过这两个元素的直线与向量的交点. 几种位置见下例:

```
\xymatrix{ & 1 \ar[r]^-{\{1000000x\}} \\ \ar[dr]_{(.2)\{2000x\}}|!{[d];[rr]}\hole & \& 1000000 \\\ 1000 \ar[r]_{\{2x\}}\ar[urr]_{>>>\{x^2\}} & \& 2000 \\ }
```



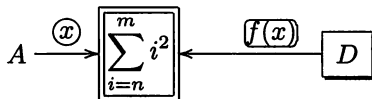
其中 `\ar[dr]_{(2)}{2000x}|!{[d];[rr]}\hole` 表示向右下方元素画的箭头有两个标记, 一个是放在右侧的 2000x, 另一个是 `\hole` 产生的洞, 修饰符 `!{[d];[rr]}` 表示洞是在 `{[d];[rr]}` 确定的方向与向量的交点处, 这个方向就是下方元素与右右元素的连线方向.

对象的修饰

矩阵元素和标记都可以进行修饰, 格式为 *修饰符{对象}, 常用的修饰符有

+	增加对象的周边空白(默认无空白)
+<尺度>	按尺度增加对象的周边空白
+=	增加成正方形
-	缩小
-<尺度>	按尺度缩小
-=	缩小成正方形
!	不居中
[l] [r] [u] [d]	对齐到左、右、上、下
[F] [F=]	方框, 双线方框(方框是长方形)
[F.] [F--]	点线框, 虚线框
[F-,] [F-:<3pt>]	阴影框, 圆角矩形框
[o]	将方框变圆

```
\xymatrix{
  A \ar[r]^-*[o][F]{x} & **[F=]{\sum_{i=n}^m i^2} \\
  & \&\& **<10pt>[F-,]{D} \ar[l]_{>>><(0.4)*[F-:<2pt>]{f(x)}}
}
```



其中 $>>><(0.4)$ 表示离箭头 $(3-1) \times \text{jot} \approx 6\text{pt}$ 处作为 (1), 箭尾作为 (0), 由此决定 (0.4) 的位置.

上面介绍了一些加框的盒子, 还有产生一般盒子的方法.

`\txt{...}` 文本盒子, 内容可以用 `\\` 换行
`\composite{... * ...}` 将 * 号两边的对象重叠合成一个对象

```
\xymatrix@1{
  A \ar@{|* \composite{\{+\}*\{\times\}}|}[rr]^-*\txt{High\\label} \\
  & \&\& B \ar@{/{*\composite{\{\texttt{Y}\}*\{-}\}}/[rr] \\
  & \&\& C }
}
```

High
label
A|*****B/YY YY YY YYC

命令 `\xymatrix@1` 用于画单行矩阵图, 效果较好.

`\txt` 命令还可以指定宽度, 并且可以用在 Xy-pic 图形命令外部, 一般格式为 `\txt<宽度>{内容}`.

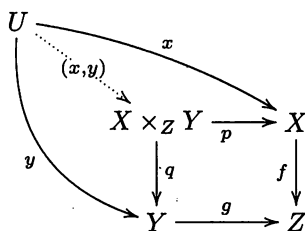
弧形向量(不能指定其他连线类型)

为使向量变成弧形, 可指定弯曲方向, 命令为

`\ar@/弯曲方向/`

`/~/` 表示向量中部向左突出, `/_/_` 表示向右突出, 弯曲的程度是 $1/8$ 圆周.
`/~1pc/` 或 `/_7mm/` 指定弯曲的程度.

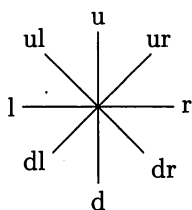
```
\xymatrix{
  U \ar@/_7mm/[ddr]_y \ar@/^/[drr]^x \ar@{.}>[dr]|-{\langle x,y \rangle} \\
  & X \times Z \times Y \ar[d]^q \ar[r]_p & X \ar[d]_f \\
  & Y \ar[r]_g & Z
```



另一种指定向量弯曲方向的命令是

`\ar@(出发方向,结束方向)`

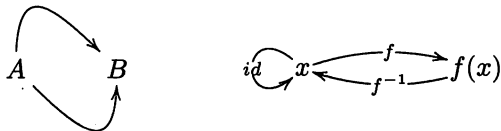
可用的方向字母及其代表的方向如下图所示, 从中心向外的方向表示出发方向, 从外指向中心的方向表示结束方向:



例如

```
\xymatrix{A \ar@(u,ul)[r] \ar@(dr,d)[r] & B} \\
\xymatrix{x \ar@(ul,dl)[ ] | \{id\} \ar@/_/[rr] | f \\
& f(x) \ar@/_/[ll] | \{f^{-1}\}}
```

的图像分别为



其他箭型

自行定义箭型的格式是

\textcircled{O} 变体{箭尾 箭杆 箭头} 或 \textcircled{O} 变体{箭头}

其中定义箭尾和箭头的符号有

< > << >> |< >| |<< >>| () / // |
|| {x} {+} {*}

定义箭杆的符号有

- -- ~ ~~ . : =

定义变体的符号有 (前两个也可只局部用于箭尾或箭头. 箭杆使用: 或 = 时, 无 2、3 变体)

~ (左侧变体) _ (右侧变体) 2 (箭杆双倍变体) 3 (箭杆三倍变体)

```
\xymatrix@1{ A \ar@{{*}\sim>|}[r] & B \ar@{<-}[r] & \; C \\ \ar@{-[{}->}[r] & D \ar@3{||-->>}[rr] & \& E }
```

$$A \bullet \rightsquigarrow B \longleftarrow C \longrightarrow D \equiv \equiv \equiv \Rightarrow E$$

输入的 \; C 是为了在元素 C 前面增加一点空白, 否则与左侧箭头太近. 上述定义的箭也可以弯曲, 如

```
\xymatrix{ A \ar@/^/@{<-}[r] & B \ar@/_/@{{*}{x}{*}}[r] & C }
```

$$A \overset{\curvearrowright}{\bullet \times \times \times \times \bullet} B$$

改变连线的起点或终点

每个连线命令都写在某个元素的左侧或右侧, 这个元素成为当前元素, 连线的起点默认是当前元素. 但也可以指定连线的起点或者改变终点.

改变连线起点: 改变起点时应显式给出起点坐标, 起点与终点之间用分号分隔. 当新的起点或终点用相对坐标时, 都是相对于当前元素的.

```
\xymatrix{ A \ar[d] \ar[d];[dr] \\ B & C \ar@<1ex>[ul] \ar@<1ex>[ul];[] }
```

$$\begin{array}{ccc} & A & \\ & \swarrow & \searrow \\ B & & C \end{array}$$

其中 @<1ex> 表示使连线向左侧移动 1ex, 若是负的尺度, 则向右移. 移动的目的是避免平行连线重叠, 上例中两条线都移动是为了看起来对称. 弯曲的连线也可平移:

```
\xymatrix@1{ A \ar@/^/[r] \ar@/^/@<-1.5mm>[r] & B }
```

$$A \curvearrowright B$$

改变连线终点: 连线的终点位置是可以改变的, 这种改变不是指使用新的终点元素坐标, 而是在原终点处附加平移向量, 使连线的终点改到新的位置. 格式为 $+<x,y>$ 或 $+方向$, 其中 $+号$ 处也可能是 $-号$, $<x,y>$ 是带有单位的尺度, 表示平移向量, 而方向只能是下列 8 种大写字母 (字母组合) 之一 (其意义是明显的):

U UR R DR D DL L UL

例如:

```
\xymatrix@1{ x \ar@/^/[r]+LU*{\circ} \ar@/_/[r]+<4mm,-1mm>*{\pi} \\ & {\pi} }
```

$$x \curvearrowright \pi$$

连线终点处的 $*{\text{对象}}$ 表示在连线终点处放置对象, 若不改变终点位置, 则对象被叠加在终点元素上. 注意, 若无 $*号$, 则对象被看成是当前元素的一部分, 相当于把连线命令写在了元素字符中间. 如

```
\xymatrix@1{ x \ar@/^/[r]*{\circ} & {\pi} }
```

得到的输出是

$$x \curvearrowright \pi$$

```
\xymatrix@1{ x \ar@/^/[r]{\circ} & {\pi} }
```

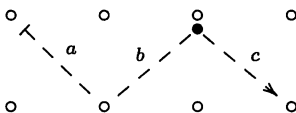
的输出是

$$x \circ \curvearrowright \pi$$

连线潜越中途目标

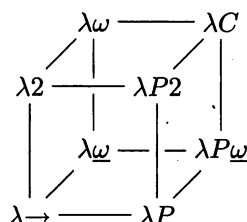
连线可以潜越中途目标, 在潜越的目标附近会留有空白, 好像连线是从目标底下通过的, 如果目标是空元素, 则潜越处会留有一小段空白. 为了表明连线潜越目标, 只需在目标前加一个单引号: '目标

```
\xymatrix{{\circ} \ar@{|-->} '[dr]^a '[rr]+D*{\bullet}^b [drrr]^c \\ & {\circ} & {\circ} & {\circ} \\ {\circ} & & & \\ {\circ} & & & }
```



```
\xymatrix@!0{ & \lambda\omega \ar@{-}[rr] \ar@{-}'[d][dd] \\
& \& \lambda C \ar@{-}[dd] \& \\
\lambda^2 \ar@{-}[ur] \ar@{-}[rr] \ar@{-}[dd] \\
& \& \lambda P^2 \ar@{-}[ur] \ar@{-}[dd] \& \\
& \& \lambda \underline{\omega} \ar@{-}'[r][rr] \\
& \& \lambda P \underline{\omega} \\
\lambda \{\to\} \ar@{-}[rr] \ar@{-}[ur] \& \lambda P \ar@{-}[ur] }
```

与上述输入对应的输出如下左图所示, 其中@! 的作用是使得图形的行列保持等距, @!0 则进一步忽略各目标的大小, 使生成的图形不受目标的影响。



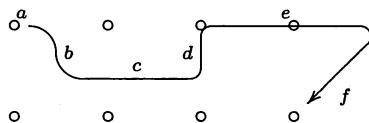
中途转弯的连线

使连线转弯的格式是: ``出发方向 目标`, 它使连线按出发方向出发向目标转 90° , 其中方向是前面介绍过的8种方向之一(小写字母). 当连续有几个转向时, 除第一个需要出发方向外, 后面转向的出发方向是显然的(就是前一个转弯后的结束方向).

``_方向 目标` 表示沿逆时针转到指定方向, `_方向 目标` 则沿顺时针转到指定方向. 此时转动角度可能大于或小于 90° .

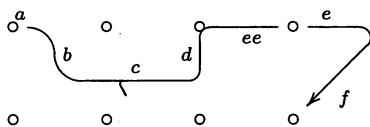
默认转动半径是10pt, 指定转动直径R(不是半径)的方法是在倒引号之后紧接写上/R, 此后的转动直径都变为R.

```
\xymatrix{ {\circ} \ar `r[d] ^a `[rr] ^b `/4pt[rr] ^c `[rrr] ^d \\
`_dl[dr] ^e [dr] ^f & {\circ} & {\circ} & {\circ} \\
{\circ} & {\circ} & {\circ} & {\circ} }
```



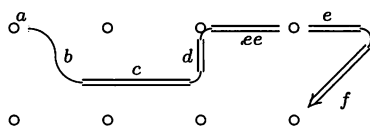
中途目标既可以是转向目标, 也可以是潜越目标:

```
\xymatrix{ {\circ} \ar `r[d] ^a `[rr] ^b `/4pt[rr] ^c `[rrr] ^d \\
'[rrr] _[ee] `_dl[dr] ^e [dr] ^f & {\circ} & {\circ} & {\circ} \\
{\circ} & {\circ} & {\circ} & {\circ} }
```



改变线型可以看出转弯长度:

```
\xymatrix{ {\circ} \ar@{>} `r[d] `a `[rr] `b `/4pt[rr] `c `[rrr]
`d '[rrr] _{ee} `dl[dr] `e [dr] `f &{\circ} &{\circ} &{\circ} \\
{\circ} &{\circ} &{\circ} &{\circ} }
```



```
\xymatrix@1{A \ar `d[r] [r] \ar@<+2pt>`d[r] `[r] [r] & B }
```



连线的一般形式 (其中一些项可以不出现):

```
\ar @/弯曲方向/ @箭型 `转向目标 标记位置 标记 '潜越目标 标记位置 标记
终点目标 标记位置 标记
```

其中各种目标既可用相对坐标也可用绝对坐标, 并可加上平移向量稍加变动. 标记可以修饰 (*修饰符{标记}). @/弯曲方向/ 可以改为 @ (出发方向, 结束方向). 终点目标前可加起点; 将起点从默认的当前元素改为指定的起点.

间隔与旋转

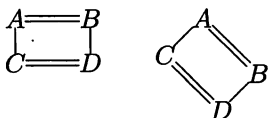
在 \xymatrix 和其后的 { 之间可以放置下列代码:

@=尺寸	设置全局间隔
@R=尺寸	设置行间隔
@C=尺寸	设置列间隔
@M=尺寸	设置元素边空 (margin)
@W=尺寸	设置元素宽度
@H=尺寸	设置元素高度
@L=尺寸	设置标记边空
@!	强制所有间隔相等
@!0	强制所有间隔相等并忽略元素大小

@!R	强制所有行间隔相等
@!C	强制所有列间隔相等
@方向	矩阵图按方向 d,u,l,r (或其组合) 旋转 (元素内容不旋转)

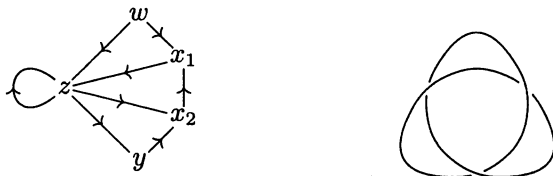
例如 $\text{\xymatrix@=Opt@M=Opt{A\&B\\C\&D}}$ 输出 $\begin{matrix} A & B \\ C & D \end{matrix}$

```
\xymatrix@M=Opt@R=1em@C=2em{
  A \ar@{=}[r] \ar@{-}[d] & B \ar@{-}[d] \\ C \ar@{=}[r] & D}
\xymatrix@dr@M=Opt@R=1em@C=2em{
  A \ar@{=}[r] \ar@{-}[d] & B \ar@{-}[d] \\ C \ar@{=}[r] & D}
```



其他特色的 xy 作图

其他还有方向图、扭结、双胞胎、多边形、网格、圆弧等特色的 xy 作图 (输入原文略去).



§ 12.5 MetaPost

功能最强的科学绘图软件可算是 John D. Hobby 开发的 MetaPost 以及下一节介绍的 PSTricks 了. 它利用 METAFONT 的基本工具把图像输出为 PostScript 的程序, 因此充分利用了这两者的强大绘图功能. MetaPost 能对点的坐标作线性运算, 能求直线或曲线间的交点, 能画参数曲线, 能对图形作仿射变换 (当然包括旋转与翻转), 能在闭合曲线所围的区域里画阴影以及着色等, 它可以调用几乎所有的 PostScript 绘图功能. 不过正因为它的强大, 要完全掌握也不容易, 尤其对用户的解析几何知识有较高要求. 本节精选了 MetaPost 的基本命令加以介绍, 相信能满足绝大多数用户的绘图需要.

MetaPost 实际上是一种作图的预处理程序, 用户先把要作的图形写成源程序, 储存成以 mp 作为扩展名的文件, 然后用 MetaPost 程序编译这个 mp 文件, 生成与源文件同名而扩展名则是一个数字的 PostScript 程序, 然后在 T_EX 源文件中加入命令读入这个 PostScript 文件, 经 T_EX 编译后生成 dvi 文件, 再通过 dvips 转化为 ps 文

件, 就可以用具有 PostScript 功能的打印机打印了, 也可以利用 GhostScript 软件在普通打印机上打印. 这就是 MetaPost 的工作流程.

长度单位

MetaPost 的默认长度单位是 PostScript point(“大点”), 记为 bp, $1\text{bp} = \frac{1}{72}\text{in} = 1.004\text{pt} = 0.352\text{mm}$. 因此点 (30, 0) 的横坐标是 30 大点, 约为 10.58 mm, 纵坐标等于 0. 除了 `em` 和 `ex` 外, MetaPost 认识 T_EX 的其他所有长度单位. 因此你也可以定义一个点 (10mm, 0). 在实际使用中不妨先定义一个长度单位 u , 例如令 $u=1\text{mm}$, 以后只要修改 u 的值, 就能起到缩放图形的作用.

连点成线

MetaPost 的基本绘图命令是

```
draw (20,20)--(0,0)--(0,30);
```

中间的 `--` 表示用直线段连接两点, 每条命令以分号 ; 结束. 如果把 `--` 改成 `..`, 就表示用 3 次样条曲线 (Bézier cubic curve) 连接各点. 此外, 我们可以在语句最后加注 `cycle` 表示闭合曲线, 其作用相当于回到第 1 个点. 现在我们可以写出第 1 个 MetaPost 源程序 (12-5-1.mp):

```
beginfig(1);
u=1mm;
draw (0,0)..(20u,13u)..(13u,30u)..(3u,23u)--(8u,16u)--cycle;
draw (30u,0)..(50u,13u)..(43u,30u)..(33u,23u)..(38u,16u)
..cycle;
endfig;
end
```

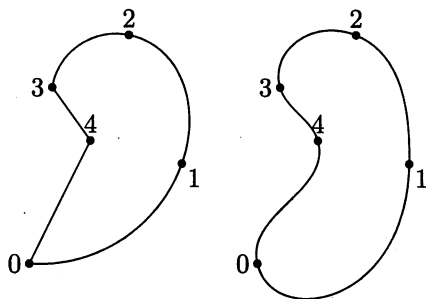
注意在没有歧义的情况下, 乘号 `*` 可以忽略, 如 $20u=20*u$. 这里 `beginfig` 与 `endfig` 之间是一个完整的图形, 一个 `mp` 文件可以包含很多个图形, 用 `beginfig` 后面括号里的数字表示图号, 因此不能重复. 最后一行的 `end` 应该是整个 `mp` 文件的结束. 文件完成后即可在命令提示符后面键入:

```
mp 12-5-1
```

回车执行后生成输出文件 12-5-1.1, 这里的后缀 1 就是图号. 然后在 T_EX 源文件中用以下命令完成嵌入图形 (见 12-5-1.tex):

```
\includegraphics{12-5-1.1}
```

再经过 `dvips` 处理生成 `ps` 文件后才能正确显示这个图形如下:



请注意这里的圆点及其编号都是另外加上去的.

画笔粗细

为了控制线条的粗细, 可以用以下命令规定画笔头的半径:

```
pickup pencircle scaled 4pt;
```

默认值是0.5bp.

3点共线与解线性方程组

利用以下命令:

```
z1 = (20u,13u);
```

可定义一个点 $z1$, 同时也定义了它的两个坐标 $x1=20u$, $y1=13u$. 反之, 给出了变量 $x1$, $y1$ 的值也就给出了点 $z1$. 不仅如此, MetaPost 还会解线性方程组, 因此命令中出现的等号实际上是代表一个方程, 与赋值号 $:=$ 不同. 从以下语句:

```
a+b=3; 2*a=b+3;
```

或

```
a+b = 2*a-b = 3;
```

都可得到 $a = 2$, $b = 1$. 此外, MetaPost 还会计算定比分点, 等式

```
z4=1/3[z3,z6];
```

相当于

$$z4 = z3 + \frac{1}{3}(z6 - z3).$$

因此

```
z20=aa[z1,z3];
```

相当于说点 $z20$, $z1$, $z3$ 共线 (注意中间省略了乘号*). MetaPost 里有一个特殊的临时变量 `whatever`, 它可取任意不确定的值, 利用它, 下面的语句:

```
z20=whatever[z1,z3]=whatever[z2,z4];
```

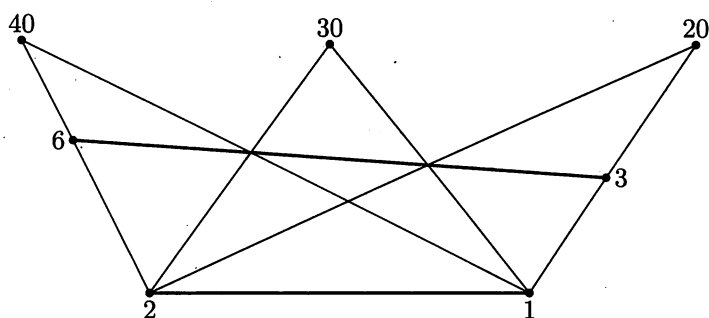
确定了 $z20$ 是直线 $z1z3$ 与 $z2z4$ 的交点. 因此, 12-5-1.mp 中的图 2:

```

beginfig(2);
u:=1mm;
z1=-z2=(25u,0);
x3=-x6=35u;
x3-2y3=x6+2y6=5u;
z4=1/3[z3,z6];
z5=2/3[z3,z6];
z20=whatever[z1,z3]=whatever[z2,z4];
z30=whatever[z1,z4]=whatever[z2,z5];
z40=whatever[z1,z5]=whatever[z2,z6];
draw z1--z20--z2--z30--z1--z40--z2;
pickup pencircle scaled 1pt;
draw z1--z2;
draw z3--z6;
endfig;

```

产生了以下图形(编号是另外加的):



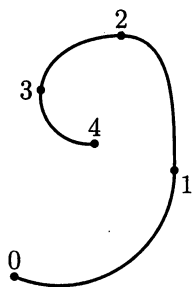
仔细研究图 2 的输入与输出可以发现要画直线图形是很容易的.

对曲线的精细控制

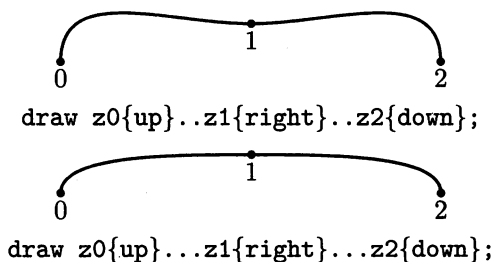
有几种方法控制样条曲线的形状. 第一种方法是规定某个节点的切线方向, 例如 $(0,0)\{\text{dir } 45\}$ 表示点 $(0,0)$ 处的切线方向应为 45° . $\{\text{dir } 0\}$ 、 $\{\text{dir } 90\}$ 、 $\{\text{dir } 180\}$ 、 $\{\text{dir } 270\}$ 可分别表示为 $\{\text{right}\}$ 、 $\{\text{up}\}$ 、 $\{\text{left}\}$ 、 $\{\text{down}\}$. 如果一个点的左右两边规定了两个不同的方向, 就说明这个点有一个角, 但这是允许的. 命令 (12-5-1.mp 的图 3):

```
draw z0..z1{up}..z2{left}..z3..z4;
```

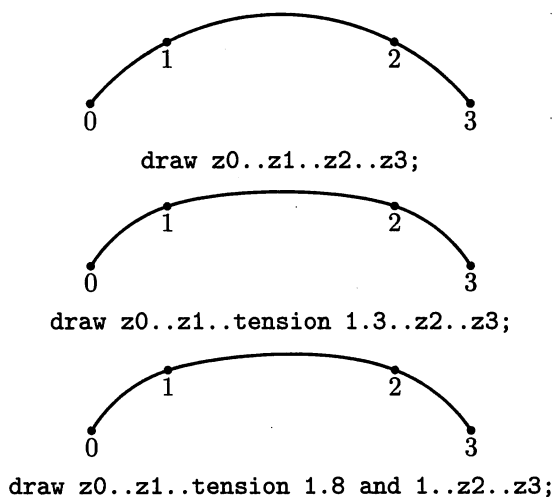
的效果如下图所示:



但是样条曲线有时会出现拐点(曲线的凹凸性发生改变的点),如果我们要求某个曲线段不含拐点,只需用3个点...代替..来连接节点,其效果见下图(12-5-1.mp的图4与5):



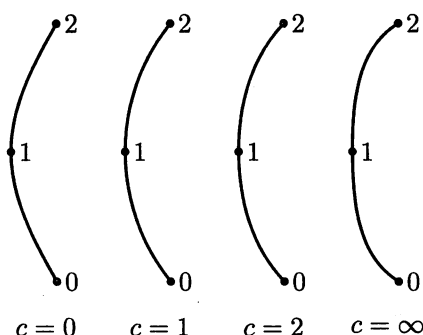
第二种方法是在两点的连线上施加一点压力,其默认值是1,例如 `tension 1.2` and `1.3` 或 `tension 1.2`. 其效果见下图(12-5-1.mp的图6、7与8):



第三种方法是在曲线的端点上规定卷曲程度,例如 `z0{curl c}`, 默认值 $c = 1$, 表示端点处的线段接近于圆弧, c 越大, 曲率越大. 命令

`draw z0{curl c}..z1..{curl c}z2;`

的效果见下图(12-5-1.mp的图9):



注意: MetaPost 用 `infinity` 表示无穷大.

第四种方法是规定控制点的坐标, 例如:

```
z1..controls (30,20) and (15,17)..z2
```

当左右坐标相等时, 可以简写成 `controls (30,20)`.

MetaPost 的数据类型

MetaPost 目前有 9 种数据类型: `numeric` (数值), `pair` (二元组), `path` (路径), `transform` (变换), `color` (色彩), `string` (字符串), `boolean` (布尔量), `picture` (图形), `pen` (画笔).

`numeric` 量在 MetaPost 内部都被化成以 $\frac{1}{65536}$ 为单位的整数, 因此它的绝对值不能超过 4096, 不过中间结果可以放宽到不超过 32768.

`pair` 类型是一对 `numeric` 量, 用于表示一个点的两个坐标. `pair` 可以做加法或减法 (相当于向量运算), 也可与一个数相乘或除以一个数.

`path` 表示一个折线或曲线, 例如: `z0--z1..z2--cycle` 等, 都是 `path` 的例子.

`transform` 类型可以表示任何一个平面仿射变换, 我们以后再介绍.

`color` 类型相当于一个数值 3 元组, 其 3 个分量分别代表红、绿、兰三种成分. 预定义的彩色有: `black=(0,0,0)`, `white=(1,1,1)`, `red=(1,0,0)`, `green=(0,1,0)`, `blue=(0,0,1)`. `color` 可以做加法或减法, 也可与一个数相乘或除以一个数. 例如 `0.4white` 就代表一种灰色. `color` 的分量必须在区间 $[0, 1]$ 内取值, 超出此区间的量会被自动截断.

`string` 类型的量可以表示成 "a string"

`boolean` 量的取值可以是 `true` 或 `false`.

`picture` 类型包含任何 MetaPost 能画出来的内容, 它可被添加或变换.

`pen` 类型规定了画笔头的大小与形状, 从而决定了图形的线条形状.

常用运算

MetaPost 的运算只有 3 种优先度, 单目运算以及 `of` 运算 (以后说明) 都是最优先的. `*`, `/`, `**` (乘方), `and` (逻辑与), `div` (取整商), `mod` (求余数), `dotprod` (求两个向量的内积) 是最优先的双目运算. 次优先的双目运算有: `+`, `-`, `++` ($a++b = \sqrt{a^2 + b^2}$), `+-` ($a+-b = \sqrt{a^2 - b^2}$), `or` (逻辑或) 等. 优先度最低的双目运算有: `<`, `<=`, `<>`, `=`, `>`, `>=`, `&` (合并字符串, 如: `"Meta" & "Post" = "MetaPost"`) 等.

注意: 根据上述原则, 同一优先度自左至右运算, 从而 $3*a**2 = (3a)^2$, $-a**2 = (-a)^2$, 与一般的编程语言不同. 在没有把握时以多加括号为好.

对于数值量, 可以使用以下函数: `abs` (取绝对值), `ceiling` (不小于变量的最小整数), `cosd` (以度为角度单位的余弦值), `floor` (不大于变量的最大整数), `mexp` ($\exp(x/256)$), `mlog` ($256 \ln(x)$), `round` (最接近变量的整数), `sind` (以度为角度单位的正弦值), `sqrt` (取平方根).

循环语句

MetaPost 的循环语句为

```
for i = a step b until c:
(语句)
endfor
```

当步长 $b = 1$ 时, 可简写成

```
for i = a upto c:
```

当步长 $b = -1$ 时, 可简写成

```
for i = a downto c:
```

也可用列表方式给出循环变量的取值, 例如以下命令:

```
draw for p=(3,1),(6,2),(7,5),(4,6): p-- endfor cycle;
```

的效果相当于

```
draw (3,1)--(6,2)--(7,5)--(4,6)--cycle;
```

标注

MetaPost 用命令 `label` 实现标注, 用后缀指示标注关于参考点的方位. 后缀可以是: `lft` (左方), `rt` (右方), `top` (上方), `bot` (下方), `ulft` (左上角), `urt` (右上角), `llft` (左下角), `lrt` (右下角), 没有后缀表示中心. 命令的格式为

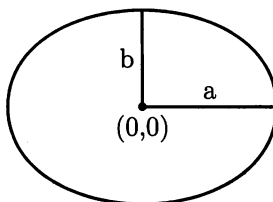
```
label.后缀(字符串或图形表达式, 二元组表达式);
```

如果把 `label` 改为 `dotlabel`, 则在参考点上标出一个黑点. 下面的例子 (12-5-1.mp 图 10):

```
beginfig(10);
a=.7in; b=.5in;
z0=(0,0);
z1--z3=(a,0);
z2--z4=(0,b);
draw z1..z2..z3..z4..cycle;
draw z1--z0--z2;
label.top("a", .5[z0,z1]);
```

```
label.lft("b", .5[z0,z2]);
dotlabel.bot("(0,0)", z0);
endfig;
```

的输出为



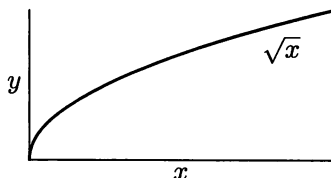
如果想要标注数学公式, 需要如下的语句:

```
btex TEX 语句 etex
```

调用 T_EX. 见下面的例子 (12-5-1.mp 的图 11):

```
beginfig(11);
numeric u;
u = 1cm;
draw (0,2u)--(0,0)--(4u,0);
pickup pencircle scaled 1pt;
draw (0,0){up}
  for i=1 upto 8: ..(i/2,sqrt(i/2))*u endfor;
label.lrt(btex  $\sqrt{x}$  etex, (3,sqrt 3)*u);
label.bot(btex  $x$  etex, (2u,0));
label.lft(btex  $y$  etex, (0,u));
endfig;
```

附带说明一下: 第2句 `numeric u;` 的作用相当于变量声明, 使得变量 u 可以重新取值, 如果没有这个声明, 由于同一文件前面的图形已经给 u 赋过值, 必须用赋值号 `:=` 代替等号, 否则就会出错. 其输出图形为



如果标注中出现汉字, 就必须读入 CJK 宏包, 这时需要使用以下结构:

```
verbatimtex TEX 语句 etex;
```

告诉 MetaPost 执行一些 T_EX 命令, 如在输入汉字标注前应该插入以下语句:

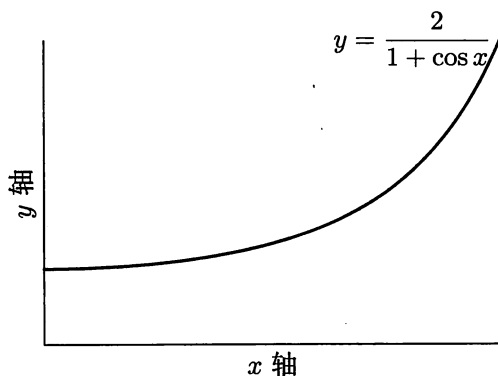
```
verbatimtex \documentclass{article}
\usepackage{CJK}
\begin{document}
\begin{CJK}{GBK}{song}
etex
```

使得 T_EX 进入 CJK 环境. 不但如此, 由于 MetaPost 执行时会自动调入 PlainT_EX, 现在我们需要的是 L^AT_EX, 因此必须在命令行里加入一个选项:

mp -tex=latex 文件名

否则会出错.

MetaPost 在执行 btex ... etex 命令后生成一个图形, 再把这个图形放在指定的位置. 因此可对这个图形施行变换, 例如加上 rotated 90 使图形逆时针旋转 90°. 为得到下图



输入为 (12-5-2.mp 的图 12)

```
verbatimtex \documentclass{article}
\usepackage{CJK} \begin{document}
\begin{CJK}{GBK}{song} etex

beginfig(12);
numeric ux, uy;
120ux=60mm; 4uy=40mm;
draw (0,4uy)--(0,0)--(120ux,0);
pickup pencircle scaled 1pt;
draw (0,uy){right}
```



```

for ix=1 upto 24:
  ..(5ix*ux, uy*2/(1+cosd 5ix))
endfor;
label.bot(btex $x$ 轴 etex, (60ux,0));
label.lft(btex $y$ 轴 etex rotated 90, (0,2uy));
label.lft(btex $\displaystyle y=\frac{2}{1+\cos x}$ etex,
  (120ux, 4uy));
endfig;

verbatimtex \end{CJK}
\end{document} etex

```

这里的两个例子也告诉我们如何画出参数曲线的图形.

填充颜色

命令:

```
fill 闭合路径表达式 withcolor 色彩表达式;
```

可以为一个闭合区域着色. 当 `withcolor` 省略时, 则用黑色填充. 与此相反,

```
unfill 闭合路径表达式;
```

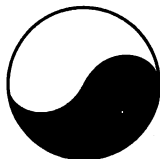
相当于用背景色 (默认为白色) 填充. 我们看下面的例子 (12-5-1.mp 图 13):

```

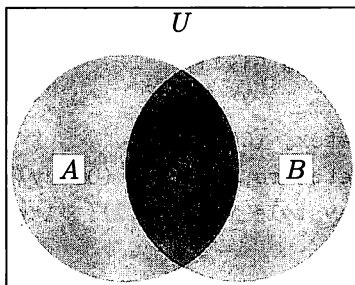
beginfig(13);
path p;
p = (-1cm,0)..(0,-1cm)..(1cm,0);
fill p{up}..(0,0){-1,-2}..{up}cycle;
draw p..(0,1cm)..cycle;
endfig;

```

其中路径 `p` 是下半圆周, 输出图形为



再看下面的图形:



其源程序为(12-5-1.mp 图 14):

```
beginfig(14);
path a, b, aa, ab;
a = fullcircle scaled 30mm;
b = a shifted (15mm,0);
aa = halfcircle scaled 30mm rotated -90;
ab = buildcycle(aa, b);
picture pa, pb;
pa = thelabel(btex $A$ etex, (-7.5mm,0));
pb = thelabel(btex $B$ etex, (22.5mm,0));
fill a withcolor .8white;
fill b withcolor .8white;
fill ab withcolor .5white;
unfill bbox pa;
draw pa;
unfill bbox pb;
draw pb;
label.top(btex $U$ etex, (7.5mm,17mm));
draw bbox currentpicture;
endfig;
```

我们作一点说明: 第3行的 `fullcircle` 是以 $(0,0)$ 为中心, 直径为 $1bp$ 的圆周, `scaled 30mm` 是一个位似变换, 使它放大为直径等于 $30mm$ 的圆周. 第4行的 `shifted (15mm,0)` 是一个平移变换, 使得路径变量 `b` 表示以 $(15mm,0)$ 为圆心的, 直径等于 $30mm$ 的圆周. 第5行的 `halfcircle` 是 `fullcircle` 的上半圆周, 经变换 `scaled 30mm rotated -90` 后, 路径变量 `aa` 表示以 $(0,0)$ 为圆心的, 直径等于 $30mm$ 的右半圆周. 第6行的函数 `buildcycle` 是取出两个路径变量相交所生成的闭合路径, 因此路径变量 `ab` 表示两个圆周相交而成的闭合曲线. 第8行的函数 `thelabel` 与 `label` 的唯一区别是: 前者把图形存储在一个图形变量 `pa` 里, 并不打印, 等遇到 `draw pa;` 命令时才打印, 而后者相当于把这两个操作一次完成. 第13行的函数 `bbox pa` 生成图形变量 `pa` 的轮廓线, 得到一个长方形的闭合路径, 与 `unfill` 联用,

可以清理出一个长方形区域供标注之用. 要注意的是, 其实 `bbox` 并不生成真正的轮廓线, 而是加了一个余量 `bboxmargin`, 其默认值等于 2 bp. 用户可以用赋值语句加以修改. 倒数第 2 行的 `bbox currentpicture` 画出了当前图形的轮廓线.

画虚线

命令:

```
draw 路径表达式 dashed 虚线模式;
```

MetaPost 中有两种预定义的虚线模式: `evenly` 是一段 3 bp 的线接以 3 bp 的空白, 以 6 bp 为一个周期; `withdots` 生成以 5 bp 为间隔的点线. 用户可以通过缩放或平移加以改变. 利用函数 `dashpattern(on 尺寸 off 尺寸 ...)` 可以自定义虚线模式. 其效果见下图 (12-5-2.mp 的图 15):

```
----- dashed evenly
- - - - - dashed evenly scaled 2
..... dashed withdots
..... dashed withdots scaled 2
———..... dashed dashpattern(on 20 off 2 on 1 off 2)
```

如果要画很多虚线, 我们可以使用命令:

```
drawoptions(dashed evenly);
```

在此以后, `draw` 命令画出的曲线都成了虚线. 要恢复默认状态, 只要再发出命令 `drawoptions()`.

画箭头

画箭头的命令是 `drawarrow` 和 `drawdblarrow`. 控制箭头形状的参数有两个: `ahlength` 规定了箭头的边长, 默认值是 4 bp; `ahangle` 规定了箭头的夹角, 默认值是 45°. 效果见下图 (12-5-1.mp 的图 16):

```
1 —————→ 2 drawarrow z1..z2
3 ←————→ 4 drawdblarrow z3..z3
5 —————→ 6 ahlength:=8; ahangle:=20;
```

仿射变换

以下是一些简单的变换:

$$\begin{aligned}
 (x,y) \text{ shifted } (a,b) &= (x+a, y+b); \\
 (x,y) \text{ rotated } \theta &= (x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta); \\
 (x,y) \text{ slanted } a &= (x+ay, y); \\
 (x,y) \text{ scaled } a &= (ax, ay); \\
 (x,y) \text{ xscaled } a &= (ax, y); \\
 (x,y) \text{ yscaled } a &= (x, ay); \\
 (x,y) \text{ zscaled } (a,b) &= (ax-by, bx+ay).
 \end{aligned}$$

其中 **zscaled** 的意义是: 把点 (x,y) 看成复数 $x+yi$, 这个变换就是用复数 $a+bi$ 相乘.

还有两个变换, 一个是关于两点 p, q 的连线的反射:

`reflectedabout(p,q)`

以及绕 p 点逆时针旋转 θ 度:

`rotatedaround(p, \theta)`

把上述变换作用在恒同变换 `identity` 上, 就能定义一个新的变换, 如:

`T = identity xscaled -1 rotated 90 shifted (1,0);`

你能想象 T 是怎样的变换吗?

给出了变换 T , 可以用以下命令:

`q = p transformed T;`

或

`q = p transformed inverse T;`

施行变换 (当然后面一条命令施行的是逆变换). 这里的 q, p 可以是图形、路径、二元组、画笔或变换中的一种.

画函数图像的 `graph` 宏包

为了使用这个宏包, 必须在 `mp` 文件的前面加入以下命令:

`input graph`

然后输入

`draw begingraph(宽度, 高度);`
`graph 的作图命令`
`endgraph;`

这里的宽度和高度是图框的尺寸, 不包含坐标刻度所占的空间.

由于 graph 环境自动根据输出画框的大小对函数图形进行缩放, 因此 MetaPost 的许多绘图命令不能直接使用, 必须改成它们的 g 形式——前面加一个字母 g, 这样的命令有: gdraw, gdotlabel, gdrawarrow, gdrawdbllarrow, gfill, glabel, 它们的用法与原来的命令一样. 只是在 glabel 命令的参考点位置可以放上 OUT 表示打印在图框的外面, 详见后面的例子.

再介绍几条新的命令:

```
setcoords(x 轴坐标, y 轴坐标);
```

用来设定坐标轴的刻度, 取值可为 linear, -linear, log, -log 之一.

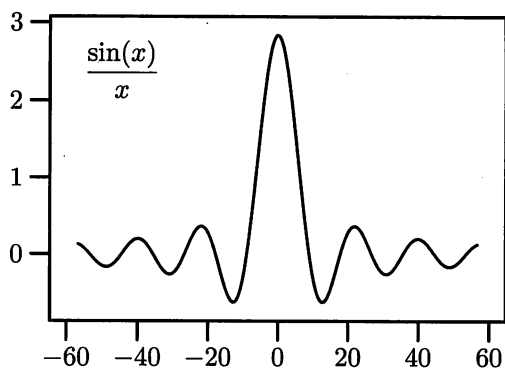
```
setrange(左下角坐标, 右上角坐标);
```

设定纵横坐标的范围.

```
augment.路径变量名(横坐标, 纵坐标);
```

在路径变量里添加一个新的节点. 读者只要仔细观察下面两个例子(12-5-2.mp 的图 17, 18, 作者根据 texmf\doc\latex\emp>manual.ps 里的例子改写), 就能知道这些命令的用法了.

```
beginfig(17);
u:=1mm;
draw begingraph(60u,40u);
  pickup pencircle scaled 1pt;
  path p;
  for x = -20 step 0.2 until -0.2:
    augment.p (x*u, sind(x*180/3.14159)/x*u);
  endfor
  augment.p (0, u);
  for x = 0.2 step 0.2 until 20:
    augment.p (x*u, sind(x*180/3.14159)/x*u);
  endfor
  glabel.lrt(btex $\displaystyle\frac{\sin(x)}{x}$ etex,
    (-20u,u));
  gdraw p;
endgraph;
endfig;
```

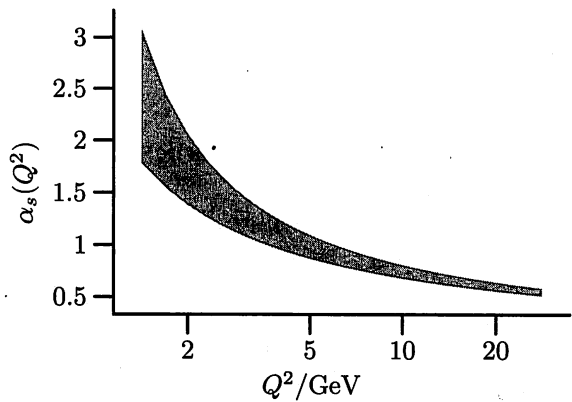


```

beginfig(18);
u:=1mm;
draw begingraph(60u,40u);
  pi = 3.14159; beta0 = 11 - 2/3*4;
  lambda1 = 0.15; lambda2 = 0.25;
  setcoords (log,linear);
  pickup pencircle scaled 1pt;
  path p[];
  for x = 0.5 step 0.1 until 10:
    augment.p1 (x*u, 4*pi/(beta0*2mlog(x/lambda1)/256)*u);
    augment.p2 (x*u, 4*pi/(beta0*2mlog(x/lambda2)/256)*u);
  endfor
  gfill p1--(reverse p2)--cycle withcolor .5white;
  glabel.lft(btex $\alpha_s(Q^2)$ etex rotated 90, OUT);
  glabel.bot(btex $Q^2/\text{textrm{GeV}}$ etex, OUT);
  frame.llft;
endgraph;
endfig;

```

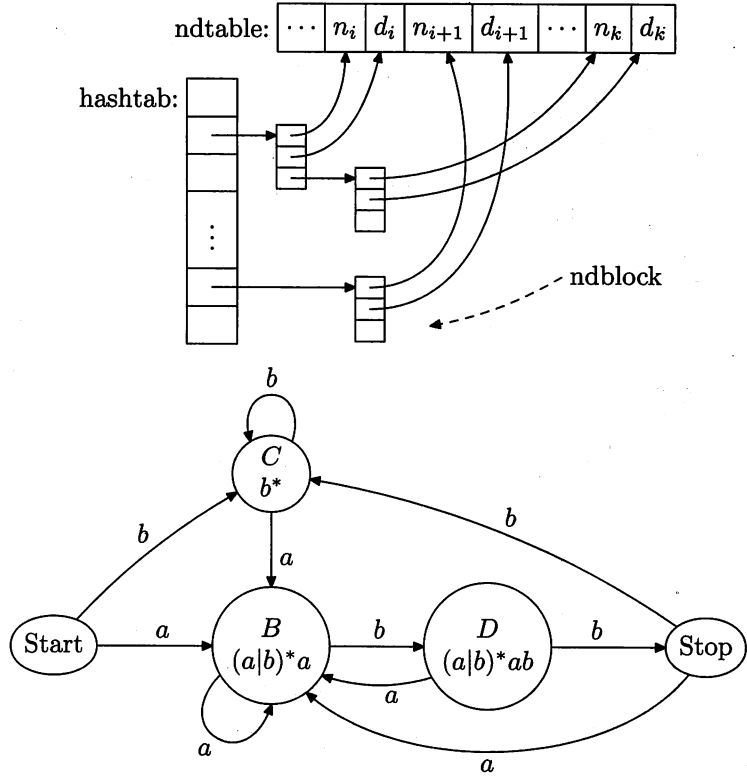
倒数第3行的命令 `frame.llrt` 的作用是只画左下角的图框。



关于 graph 宏包的更详细的用法介绍, 请参看 `texmf\doc\metapost\base\mpgraph.pdf`.

画流程图的 boxes 宏包

MetaPost 还有一个强有力的 boxes 宏包, 它的技巧要求也更高一些, 我们这里只给出用这个宏包画的两个流程图, 供读者欣赏. 本节的例子大都来自 `texmf\doc\metapost\base\manfig.mp`, 下面的两个流程图就是其中的图 49 和 52.



对MetaPost有兴趣的读者可以参看用户手册 `texmf\doc\metapost\base\mpman.pdf`, 本节的内容都是取材于此手册.

§ 12.6 PSTricks

Timothy van Zandt 开发的 PSTricks 也是直接利用 PostScript 图形功能的宏包, 内容丰富, 功能强大. 我们这里只能摘要介绍, 详情可到 PSTricks 的网站查阅: <http://tug.org/PSTricks>. 本节的大部分内容以及例题均取自印度 T_EX 用户协会制作的 L^AT_EX 在线教材以及其他说明书.

为了使用 PSTricks, 先在导言区加入命令 `\usepackage{pstricks}`. 为简单起见, 以下我们将使用简称 PST 代替 PSTricks. 为了节省篇幅, 本节主要介绍命令的用法, 把大部分附图略去. 在 12-6-1.tex 至 12-6-10.tex 内收录了本节的例, 此外还收集了 PST 手册中一些例题的源代码, 并编了图号, 读者只要用 L^AT_EX+dvips 编译一下, 再与源文件对照, 就能学到很多技巧.

PST 同样是基于直角坐标系的, 它把原点取为当前的位置, 以向右、向上为正方向. 并且把默认的单位长度规定为 1 cm. 改变单位长度的命令是

```
\psset{unit=长度}
```

`unit` 定义了单位长度, 是最基本的参数. 它可以出现在绝大多数命令的选项中, 例如设 `unit=0.6cm` 等于把这个图缩小到原始大小的 60%. 类似的还有 `xunit` 与 `yunit`, 分别规定了水平或竖直方向上的单位长度. `runit` 规定了圆弧的半径或直径的单位长度. 它们都可以用 `\psset` 命令或在其他命令的选项中用 `unit=长度` 的形式改变它的值. `\psset` 命令中还可以设定其他参数, 并且这种设定是全局性的. 而出现在其他命令选项中的设定则只对这个命令的作用范围有效.

用 PST 画直线或折线

PST 的画直线或折线的命令是

```
\psline[选项]{箭型}(x_1,y_1)(x_2,y_2)...\dots(x_n,y_n)
```

命令中的选项与箭型都可以不出现, 这时就以默认的线条画实线. 当点数 n 等于 1 时, 就以原点 (0,0) 作为默认的起点. 例如输入

前一句结束. `\psline(2,0)\psline(0,0.5)`

得到的输出为

前一句结束. 

而且 PST 命令都是不占位置的, 如下例所示: 输入

前一句结束. `\psline(0,-0.1)(2,-0.1)\psline(0,0.5)` 下一句开始

得到的输出为

前一句结束。| 下一句开始

请注意，有一小段被盖住了。
如果我们不希望发生这种叠置的现象，可以利用以下的环境建立一个盒子：

```
\begin{pspicture}(x_0,y_0)(x_1,y_1)
.....
\end{pspicture}
```

其中 (x_0,y_0) 与 (x_1,y_1) 分别是盒子的左下角及右上角的坐标，当只出现一个坐标时，表示 $(x_0,y_0) = (0,0)$ 。例如输入
















```
前一句结束.\begin{pspicture}(-0.2,0.1)(0.5,0.6)
\psline(2,0)\psline(0,0.5) \end{pspicture} 下一句开始
```

得到的输出为

前一句结束。| 下一句开始

pspicture 的盒子的左下角放在行的基线上。不过我们可以用添加选项的方法使其下沉或升高，例如 `\begin{pspicture}[shift=-0.5cm]` 就表示把盒子的左下角放在基线下方 0.5 cm 处，取正值就是放在上方。

现在我们回过头来介绍 \psline 的参数。箭型可以是

输 入	样 式	说 明
-		实线
<->		箭头
>-<		逆向箭头
<<->>		双重箭头
>>-<<		双重逆向箭头
-		T 形条，限制在端点坐标之内
* - *		T 形条，以端点坐标作为中心
[-]		方括号
(-)		圆括号
o - o		圆圈，以端点坐标为圆心
* - *		圆盘，以端点坐标为圆心
** - **		圆盘，限制于端点之内
C - C		外突的圆头
cc - cc		受限于端点坐标的圆头
C - C		外突的方头

当然线条的两端可以取各种不同类型的组合。也可以在选项中用 `arrows=箭型` 的方式规定箭型。

选项可以是

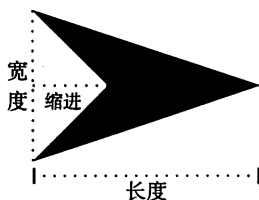
<code>linecolor=色彩</code>	线条色彩可以是 red, green, magenta, yellow, blue, cyan
<code>linestyle=类型</code>	线条类型可以是 dashed (虚线), dotted (点线)
<code>linewidth=宽度</code>	默认值是 0.8 pt
<code>doubleline=true</code>	箭身为双线
<code>doublesep=宽度</code>	当箭身为双线时, 中间的间隙
<code>linearc=长度</code>	折线拐弯时的弧线半径
<code>showpoints=true</code>	显示各节点
<code>arrows=箭型</code>	规定箭型

其中 `doublesep` 的默认值是 $1.25 \times \text{linewidth}$.

为了定义各种尺寸, 可输入以下参数, 表格中列举的都是默认值:

参 数	值	说 明
<code>dash=5pt 3pt</code>	线段长 5 pt, 间隙 3pt	虚线 (dashed) 的参数
<code>dotsep=3pt</code>	间隙 3pt	点线 (dotted) 的参数
<code>dotsize=0.5pt 5</code>	$5 \times \text{linewidth} + 0.5\text{pt}$	圆的直径
<code>tbarsize=2pt 5</code>	$5 \times \text{linewidth} + 2\text{pt}$	T 形条、方括号或圆括号的宽度
<code>bracketlength=0.15</code>	$0.15 \times \text{宽度}$	方括号的长度
<code>rbracketlength=0.15</code>	$0.15 \times \text{宽度}$	圆括号的长度
<code>arrowsize=2pt 3</code> <code>arrowlength=1.4</code> <code>arrowinset=0.4</code>	宽度 = $3 \times \text{linewidth} + 2\text{pt}$ 长度 = $1.4 \times \text{宽度}$ 缩进 = $0.4 \times \text{长度}$	箭头形状

箭头形状的参数含义如下图所示:



读者可参看 12-6-1.tex 中的图 1 至 6 了解这些参数的含义.

画闭合折线有下面的命令:

```
\pspolygon[选项](x_1,y_1)(x_2,y_2)\dots(x_n,y_n)
\pspolygon*(x_1,y_1)(x_2,y_2)\dots(x_n,y_n)
```

其中带星号命令画的是实心多边形。画矩形也有一个命令：

```
\psframe[选项](x_1,y_1)(x_2,y_2)
```

相应地也有一个带星号的命令，表示画实心矩形。如果要画圆角矩形，可以加入选项 `[framearc=数]`，这样矩形圆角的直径等于数 \times 矩形的短边。这样得到的圆弧直径是相对的。如果要规定与矩形大小无关的绝对的值，可设选项为 `[cornersize=absolute,lineararc=数]`。例见 12-6-1.tex 中的图 7、8。

类似地有画等腰三角形和菱形的命令：

```
\pstriangle(x,y)(b,h)
\psdiamond(x,y)(d_1,d_2)
```

此三角形的水平底边长为 b ，中点在 (x,y) ，高为 h 。菱形的中心在 (x,y) ，水平与竖直对角线分别长 $2d_1$ 与 $2d_2$ 。它们也有相应的星号命令。例见 12-6-1.tex 中的图 9。

用 PST 画简单曲线

首先补充一个画点的命令：

```
\psdots[选项](x,y)
```

画圆的命令是：

```
\pscircle[选项](x,y){r}
```

圆心为 (x,y) ，半径是 r 。其选项与前面的直线相同。相应的星号命令画一个实心圆。例见 12-6-2.tex 的图 10。

画圆弧的命令是：

```
\psarc[选项](x,y){r}{deg_1}{deg_2}
\psarcn[选项](x,y){r}{deg_1}{deg_2}
```

圆心为 (x,y) ，半径是 r ， \deg_1 、 \deg_2 分别是起始与结束的角度。`\psarc` 是按逆时针方向画弧，而 `\psarcn` 则是按顺时针方向画弧。如果选项取成 `[showpoints=true]`，那么圆心与相应的半径都会用虚线画出。相应的星号命令画一个实心弓形。例见 12-6-2.tex 的图 11 至 13。

画空心扇形的命令是

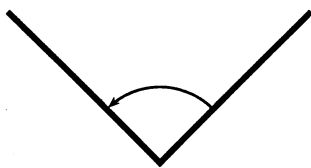
```
\pswedge[选项](x,y){r}{deg_1}{deg_2}
```

其参数的含义同圆弧. 相应的星号命令画一个实心扇形. 例见 12-6-2.tex 的图 14.

如果你要画一个角, 并在角的内部画一个带箭头的弧线. 当角的两条边是粗实线时, 箭头有可能被遮没. 这时可使用选项 `arcsepA` 与 `arcsepB`. 如果加入选项 `[arcsepB=2pt]`, 其意义是使得弧线的终点与宽度为 2pt 的直线恰好接触. `arcsepA` 有类似的含义. `arcsep` 则同时设定两端的值. 例如输入 (见 12-6-2.tex 的图 15)

```
\begin{pspicture}(-2,0)(2,2)
  \psline[linewidth=2pt](2,2)(0,0)(-2,2)
  \psarc[arcsepB=2pt]{->}(0,0)1{45}{135}
\end{pspicture}
```

可以得到



还有画椭圆和抛物线的命令:

```
\psellipse[选项](x,y)(a,b)
\parabola[选项](x_0,y_0)(x_1,y_1)
```

分别画一个中心在 (x,y) , 水平半轴为 a , 竖直半轴为 b 的椭圆, 以及起始点为 (x_0,y_0) , 极值点为 (x_1,y_1) 的抛物线. 相应的星号命令画一个实心椭圆以及抛物弓形. 例见 12-6-2.tex 的图 16 及 17.

彩色

PSTricks 会自动调用宏包 `xcolor` 使得图形中能使用各种彩色. 已经定义好的色彩有 `red`, `green`, `blue`, `cyan`, `magenta`, `yellow`, `black`, `white`, `orange`, `violet`, `purple`, `brown`, `pink`, `olive`, `darkgray`, `gray`, `lightgray`. 如果使用带选项的命令 `\usepackage[dvipsnames]{pstricks}`, 就能使用 dvips 定义的 68 种颜色 (参见 § 13.1). 我们可以用选项 `linecolor` 来定义线条或者填充的颜色. 参见 12-6-3.tex 的图 18, 19.

对于空心闭合图形也可以使用选项

```
\psframe[fillstyle=solid,fillcolor=yellow](0,0)(3,3)
```

使内部填充颜色. 观察 12-6-3.tex 的图 20, 可以发现边界是黑色, 内部则是黄色. 如果你希望边界与内容使用同一种颜色, 可以使用选项 `linestyle=none`, 见 12-6-3.tex 的图 21, 这时得到的图形与带星号的命令没有什么区别. 当然你也可以用 `linecolor` 定义边线的颜色, 见 12-6-3.tex 的图 22.

选项 `fillstyle`= 后面的参数除了 `solid` 外, 还可以取 `vlines`, `hlines` 与 `crosshatch`, 即分别用竖直线、水平线或交叉线填充. 参数 `hatchwidth` 是填充线的宽度, 默认值是 0.8pt. 参数 `hatchsep` 是填充线的间隔. 参数 `hatchwidththinc` 是填充线宽度的增量, 默认值 0. 参数 `hatchsepinc` 是填充线间隔的增量, 默认值 0. 参数 `hatchcolor` 是填充线的颜色. `fillcolor` 则是背景色. 还有一个参数 `hatchangle` 控制填充线的方向, 默认值是 45. 参见 12-6-3.tex 的图 23 至 25.

自定义色彩可以使用以下命令 (参见 § 13.1):

```
\definecolor{色彩名}{模式}{数据}
```

最后介绍色彩的渐变, 为了能使用此效果, 需要在导言区加入命令 `\usepackage{pst-grad}`. 然后要设定选项 `fillstyle=gradient`, 并且设定起始颜色 `gradbegin=色彩`, 终结颜色 `gradend=色彩`. 请比较 12-6-3.tex 的图 26 (没有渐变) 与图 27 的差异. 注意观察图 27, 天空的颜色自上而下由浅蓝色渐变至淡粉红色, 但最后又从淡粉红色变回浅蓝色. 下面的草地也有同样的现象. 这是由参数 `gradmidpoint` 控制的: 它的取值范围从 0 到 1, 默认值是 0.9. 图 28 中它的值取为 1, 读者可观察其效果. 还有一个控制渐变的参数是角度 `gradangle`, 请观察图 28a 比较不同角度的效果.

对边框的装饰

对边框的装饰就是改变线条的类型. 以前已介绍过用 `doubleline=true` 设置双线条, 用 `doublesep` 控制双线间隙, 用 `linecolor` 控制线条颜色. 其效果可参见 12-6-4.tex 的图 29, 30. 还可以用 `doublecolor` 控制线间空隙的颜色, 见 12-6-4.tex 的图 31. 图 32 与 33 展示了双线条与点线结合产生的效果.

选项 `dimen` 控制边框线与图形边界的位置关系. 其取值可以为 `outer`, `middle`, `inner`, 分别规定边框线的外侧、中心线或内侧与图形的定义边界一致. 默认值是 `outer`. 12-6-4.tex 的图 34, 35 显示了它们的差别.

参数 `border=宽度` 是用来控制边界的宽度的, 其默认值是 0pt. 同时可以用 `bordercolor` 规定其颜色. 12-6-4.tex 的图 36 至 39 显示了这些参数的使用效果.

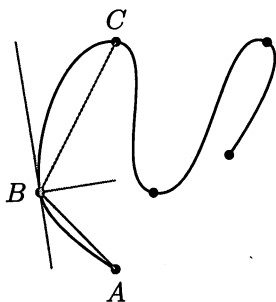
选项 `shadow=true` 给边框加上阴影. 阴影的颜色可用 `shadowcolor` 设定, 默认值是 `darkgrey`. 阴影的宽度由 `shadowsize` 确定, 默认值是 3pt. `shadowangle` 规定了阴影的方向, 默认值是 -45. 通过适当设置阴影的参数可以产生“月食”的效果, 参见 12-6-4.tex 的图 40 至 44.

画曲线

画一条通过指定点的曲线的命令是

```
\pscurve[选项]{箭型}(x_1,y_1)(x_2,y_2)\dots(x_n,y_n)
\psccurve[选项](x_1,y_1)(x_2,y_2)\dots(x_n,y_n)
\psecurve[选项]{箭型}(x_1,y_1)(x_2,y_2)\dots(x_n,y_n)
```

关于直线的选项都可用于曲线. 请注意第二条命令多一个 `c`, 表示画闭曲线. 而第三条命令多一个 `e`, 表示首尾两点 (x_1, y_1) 与 (x_n, y_n) 不被画出来, 这两个点的作用是控制曲线的形状. 见 12-6-5.tex 的图 45 至 50. 其中图 49 与图 50 显示了变化首尾两点对曲线形状的影响. 曲线的绘图原理是这样的: 设 A, B, C 是 3 个相继的点, 则 B 点的曲线段的法线应该是 $\angle ABC$ 的角平分线, 如下图所示.



对于高级用户来说, 如果想更随心所欲地控制曲线的形状, 那么可以使用参数 `curvature`, 它对应 3 个 -1 到 2 之间的数, 默认值为

```
curvature=1 0.1 0
```

其中第一个值是张力的大小, 值越小, 张力越大. 12-6-5.tex 的图 51 显示了 `curvature` 的第一个值的不同取法对曲线形状的影响.

`curvature` 的第二个数也是与松紧性有关的, 不过它仅作用于某些点对之间的曲线段. 设 A, B, C 是 3 个相继的点, 如果 $\angle ABC < 45^\circ$, 那么第二个数越小, 在 B 点附近的张力越大; 而若 $\angle ABC > 45^\circ$, 那么效果恰相反, 也即第二个数越小, 在 B 点附近的张力越小. 参见 12-6-5.tex 的图 52, 注意其中只有第二个点的夹角大于 45° . 因此当第二个参数减小时, 第二个点附近的曲线段越来越紧, 其他点附近的曲线段则越来越松.

`curvature` 的第三个数控制了曲线在每个点的斜率. 设 A, B, C 是 3 个相继的点, 当这个数等于 0 时, 曲线在 B 的切线垂直于 $\angle ABC < 45^\circ$ 的角平分线; 而取值 -1 会使 B 点的切线平行于 AC . 12-6-5.tex 的图 53 显示了这一情况. 可以看出当这个参数减小时, B 点的切线按逆时针方向转动.

PST 还有一个画 Bézier 曲线的命令:

```
\psbezier[选项]{箭型}(x_1, y_1)(x_2, y_2)(x_3, y_3)(x_4, y_4)
```

画一条连接 (x_1, y_1) , (x_4, y_4) 的曲线, 以 (x_2, y_2) , (x_3, y_3) 作为控制点. 关于其原理请读者参看 12-6-5.tex 的图 54.

坐标系

PST 里有一个画坐标网格的命令:

`\psgrid[选项](x_0,y_0)(x_1,y_1)(x_2,y_2)`

这个网格以 (x_1,y_1) , (x_2,y_2) 作为对角顶点, 而 (x_0,y_0) 用于标注坐标, 也就是以 (x_0,y_0) 作为坐标标注的“原点”, 参见 12-6-6.tex 的图 55. 当参数只有两个点时, 相当于第一个点重复使用两次, 即这两个点是对角顶点, 第一个点又用作标注的起点. 参见 12-6-6.tex 的图 56, 57, 请读者比较两者的异同. 如果出现在 `pspicture` 环境里, 那么不带参数的 `psgrid` 命令继承了 `pspicture` 的参数, 参见 12-6-6.tex 的图 58.

`\psgrid` 的选项见下表:

选 项	说 明	默认值
<code>subgriddiv</code>	主网格划分数	5
<code>gridwidth</code>	主网格线的宽度	0.8 pt
<code>subgridwidth</code>	次网格线的宽度	0.4 pt
<code>griddots</code>	当这个数大于 0 时, 主网格是点线, 每格有这么多个点	0
<code>subgriddots</code>	当这个数大于 0 时, 次网格是点线, 每格有这么多个点	0
<code>gridlabels</code>	坐标标注的字体大小	10 pt
<code>gridcolor</code>	主网格线的颜色	black
<code>subgridcolor</code>	次网格线的颜色	black
<code>gridlabelcolor</code>	网格标注的颜色	black

如果要使用极坐标, 则可以用以下命令:

`\SpecialCoor`

在这个命令以后就可以使用极坐标了, 点的极坐标记为 $(r;a)$, 注意中间用分号分开, 以与直角坐标区分. 角度的单位是度, 为了分圆的方便, 有一条命令

`\degrees[数]`

例如 `\degrees[7]` 使得以后出现的角度以 $360/7$ 度为单位. 这样可便于画正多边形, 参看 12-6-6.tex 的图 59 至 61. 当然也有命令

`\radians`

使角度单位取成弧度.
在命令选项中还可以使用

`origin={x,y}`

使得新坐标系的原点移到 (x, y) . 请注意命令中的坐标放在花括号里. 这相当于对图形做一个平移. 参看 12-6-6.tex 的图 62. 类似地还有一个交换 x, y 轴的选项 `swapaxes=true`. 参看 12-6-6.tex 的图 63.

放置或移动绘画对象

命令

```
\rput(坐标){对象}
```

把圆括号内的坐标所确定的点作为花括号内对象的原点, 相当于把对象作平移. 请参看 12-6-7.tex 的图 64. 命令

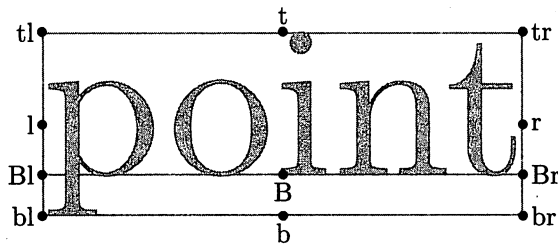
```
\rput{角度}(坐标){对象}
```

还使对象旋转一个指定的角度. 请参看 12-6-7.tex 的图 65.

如果对象是文本, 就等于把文本的中心点放在这个参考点上. 因此这条命令对于图上注字特别有用. 对这条命令还可以加一个参数

```
\rput[位置]{角度}(坐标){对象}
```

这里的位置是指参考点关于文本盒子的位置, 它的取值及意义可从下图看出:



12-6-7.tex 的图 66 至 70 显示了这些参数的效果. 图 71 则是上述命令在画统计图中的应用, 请注意其中用到了 `\rput` 的带星号形式.

以下命令更适合于为图形标注:

```
\uput{长度}[方向角]{旋转角}(坐标){对象}
```

根据这条命令, 标注文字放在参考点的由方向角规定的方向上, 与参考点的距离等于长度. 其中长度就是参数 `labelsep` 的值, 其默认值是 5 pt. 可以用 `\psset` 为 `labelsep` 赋值, 而且命令 `\uput` 中的 {长度} 可以省去. 此外这里的方向角与旋转角的角度可以用下表的字母代替. 请参看 12-6-7.tex 的图 72 至 74.

角度	字母	意义	角度	字母	意义
0	r	右方	45	ur	右上方
90	u	上方	135	ul	左上方
180	l	左方	225	dl	左下方
270	d	下方	315	dr	右下方

用方程画曲线

为了画函数图像, 首先要利用命令 `\usepackage{pst-plot}` 调用宏包. 其次要学会把函数写成 PostScript 命令的形式. 下表列出了 PostScript 的一些算子. 我们将通过例子学会把函数写成 PostScript 的语言.

算 子	意 义	语 法	例	
			输 入	结 果
add	两数之和	数1 数2 add	7 2 add	9
sub	两数之差	数1 数2 sub	7 2 sub	5
mul	两数之积	数1 数2 mul	7 2 mul	14
div	两数之商	数1 数2 div	7 2 div	3.5
exp	数的指数函数	数1 数2 exp	7 2 exp	49
idiv	整数相除之整商	数1 数2 idiv	7 2 idiv	3
mod	整数相除之余数	数1 数2 mod	7 2 mod	1
sqrt	数的平方根	数 sqrt	16 sqrt	4
neg	相反数	数 neg	7 neg	-7
abs	绝对值	数 abs	-7 abs	7
ceiling	向上取整	数 ceiling	7.6 ceiling	8
floor	向下取整	数 floor	7.6 floor	7
round	四舍五入	数 round	7.6 round	8
			7.2 round	7
sin	正弦(度为单位)	数 sin	30 sin	0.5
cos	余弦(度为单位)	数 cos	60 cos	0.5
atan	反正切(度为单位)	数 atan	1 atan	45
ln	自然对数	数 ln	2.71828182 ln	1
log	常用对数	数 log	100 log	2

还有一个重复算符 dup, 例如 `x dup mul` 就是计算 $x \times x = x^2$.

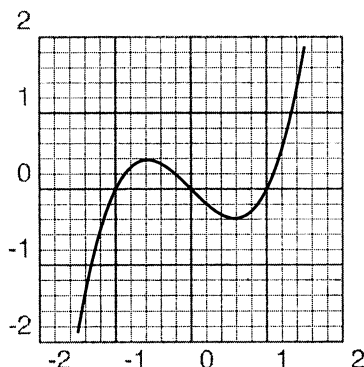
画函数图像的命令是

\psplot[选项]{下界}{上界}{函数式}

其中下界、上界规定了变量 x 的变化范围, 函数式应该用 PostScript 的语言表出. 选项中最重要的是 `plotstyle` 与 `plotpoints`. 当 `plotstyle` 等于 `curve` 时, `\psplot` 在 `plotpoints` 个点上计算函数值, 再用 `\pscurve` 画成曲线. `plotpoints` 的默认值是 50. `plotstyle` 也可以取成 `ccurve`、`ecurve`、`line`、`polygon`, 也就是分别用 `\psccurve`、`\psecurve`、`\psline`、`\pspolygon` 连点成线. 如果 `plotstyle=dots`, 则这些点不被连成线, 这时可用 `dotstyle` 规定点的形状.

下面我们画函数 $y = x^3 - x$ 在 $-1.5 \leq x \leq 1.5$ 内的图像. 使用的命令是

```
\psplot[plotstyle=curve]{-1.5}{1.5}{x 3 exp x sub}
```



更多例子请参看 12-6-8.tex 的图 75 至 77. 图 78 画出了正弦函数的图像, 由于自变量的单位应是弧度, 而 PostScript 中用的是度, 因此使用了选项 `xunit=0.0174` 以解决此问题.

如果要画参数曲线, 可使用以下命令:

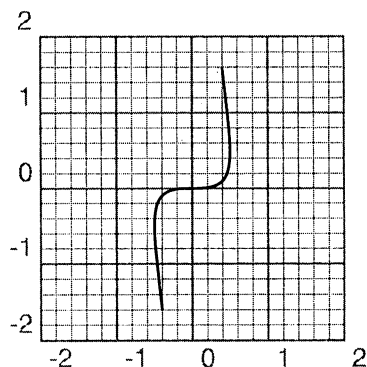
```
\parametricplot[选项]{下界}{上界}{函数式}
```

例如为了画参数曲线

$$\begin{cases} x = \frac{t}{1+t^2} \\ y = \frac{t^3}{1+t^2} \end{cases} \quad -2 \leq t \leq 2$$

的图像, 我们可以使用以下命令:

```
\parametricplot[plotstyle=curve]{-2}{2}
{t t 2 exp 1 add div t 3 exp t 2 exp 1 add div}
```



请参看 12-6-8.tex 的图 79. 图 80 则是反正弦函数的图像.

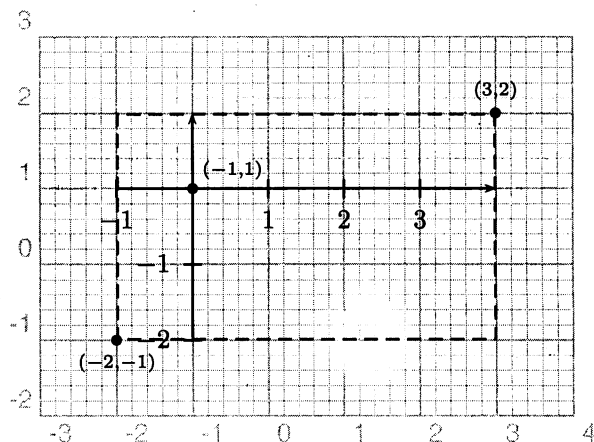
PST 还提供了画坐标架的命令:

```
\psaxes[选项]{箭头样式}(x_0,y_0)(x_1,y_1)(x_2,y_2)
```

其中坐标架的范围是由两个对顶点 (x_1, y_1) 、 (x_2, y_2) 确定的, (x_0, y_0) 则是两个坐标轴的交点. 如果只给出两个点, 就认为第一个点重复使用两次, 如果只给一个点, 就认为前面两个点都是坐标原点 $(0, 0)$. 当 {箭头样式} 是 $\{->\}$ 时, 两个坐标轴的正方向都会画上箭头. 例如命令

```
\psaxes{->}(-1,1)(-2,-1)(3,2)
```

得到的坐标架为 (亦见 12-6-8.tex 的图 81 至 84).



`\psaxes` 可以带选项 `ticks` 和 `labels`, 它们可取 4 个值: `all` (默认值, 在两个轴上都画短线或注坐标值), `x` (仅在 x 轴上画短线或注坐标值), `y` (仅在 y 轴上画短线或注坐标值), `none` (不画短线或注坐标值). 其效果见 12-6-8.tex 的图 85. 选项 `tickstyle` 可以控制短线的位置, 默认值是 `full`, 向两侧伸展; `top` 仅画在标

注的另一侧, `bottom` 则画在标注的同侧. `ticksize` 规定了短线的长度, 默认值是 3pt. 其效果见 12-6-8.tex 的图 86, 87. 如果你不要用 (0,0) 标注原点, 可设选项 `showorigin=false` (默认值是 `true`). 见图 88.

当图像只出现在一个象限里的话, 标注自然出现在象限的外侧, 如图 89. 如果出现在几个象限里, 标注将出现在 `psaxes` 中先出现的角点的象限内部, 见图 90 至 93.

参数 `labelsep` 控制了标注与轴之间的距离. 此外, 水平标注的特性 (如颜色、字体大小等) 由命令 `\pshlabel` 控制, 类似地, `\psvlabel` 控制了竖直标注的特性. 它们可用以下方式改变定义:

```
\renewcommand{\pshlabel}[1]{\color{red}#1}
```

例见 12-6-8.tex 的图 94.

但是往往图像的原点不一定是 (0,0), 这时可用以下命令规定标注的方式:

```
\psaxes[Ox=1994,Oy=0,Dx=1,Dy=10,dx=2,dy=1](10.5,7.5)
```

上述选项的含义是: 原点的 x 坐标标注为 1994, 增量是 1, 两个标注间相距 $2 \times \text{\psxunit}$; 原点的 y 坐标标注为 0, 增量是 10, 两个标注间相距 $1 \times \text{\psyunit}$. 这里 `\psxunit` 和 `\psyunit` 是 \TeX 的长度, 分别记录了 `xunit` 和 `yunit` 的值. 例见 12-6-8.tex 的图 95. 上述各数也可以是小数, 不过 `Ox` 与 `Dx` 应该有相同的小数位数 (整数除外). 参见图 96.

有时需要从文件里提取数据画出连线图, 这时可使用命令:

```
\fileplot[选项]{文件名}
```

文件名所指的文件应该与被编译的 \TeX 文件在同一个目录里, 而且应该是一个文本文件, 其中只包括数据, 即一些点的坐标对, 数与数的分隔符可以是: 空格、回车、逗号、圆括号 () 与 花括号 { }. 这条命令的 `plotstyle` 只能是: `line` (默认值), `polygon` 与 `dots`. 而且不能有 `arrows` (箭头), `linearc` 与 `showpoints`. 如果你想要把输入数据对应的点画出来, 那么你必须再做一次 `\fileplot`, 以 `plotstyle=dots` 作为选项. 参见 12-6-8.tex 的图 97, 98.

有一个类似的命令

```
\dataplot[选项]{命令}
```

这里的命令是用 `\readdata` 定义的读入数据的命令. 这个命令的选项可以使用 `curve` 以及 `showpoints`. 具体用法请参见 12-6-8.tex 的图 99. 还有一个命令

```
\listplot[选项]{数据}
```

其中数据的分隔符只能是空格或回车. 用法请参见 12-6-8.tex 的图 100.

带边框的文字以及蛇行的文字

使用以下命令可在文字的外面加上边框:

```
\psframebox[选项]{文字内容}
\psshadowbox[选项]{文字内容}
\pscircularbox[选项]{文字内容}
\psovalbox[选项]{文字内容}
```

上述命令分别生成长方形、带阴影的长方形、圆形或椭圆形的边框. 例见 12-6-9.tex 的图 101.

以下命令

```
\pstextpath[位置](x,y){图形对象}{文字}
```

可以把一段文字安放在图形对象描出的轨迹上. 位置的取值可以是 l (从轨迹起首开始安放, 这是默认值)、c (放在轨迹的中间) 或 r (放在轨迹尾部). (x, y) 规定文字与轨迹间的距离, 在水平方向的偏离距离是 x , 竖直方向偏离 y . 不过为了使用这条命令必须在导言区加入命令 `\usepackage{pst-text}`. 例见 12-6-9.tex 的图 102、103.

重复执行

以下命令是重复执行 `\rput`:

```
\multirput[位置]{角度}(起始坐标)(坐标增量){重复次数}{对象}
```

例见 12-6-9.tex 的图 104. 另一个重复执行作图的命令是

```
\multips{角度}(起始坐标)(坐标增量){重复次数}{图形对象}
```

例见 12-6-9.tex 的图 105.

变换

有以下一些变换命令:

```
\psrotateleft{对象}
\psrotateright{对象}
\psrotatedown{对象}
\psscalebox{水平倍数, 竖直倍数}{对象}
\psscaleboxto(水平大小, 竖直大小){对象}
```

前3条命令分别把对象向左、向右旋转 90° 或者旋转 180° . `\psscalebox`是缩放指定的倍数, 如果只给出一个参数, 就认为在水平与竖直方向缩放相同的倍数. `\psscaleboxto`是缩放到指定的大小. 例见12-6-9.tex的图106.

如果在导言区加入命令`\usepackage{pst-3d}`, 还可以使用以下使图形倾斜的命令:

```
\pstilt{角度}{对象}
\psshadow{文字内容}
```

例见12-6-9.tex的图107.

螺旋形曲线

在导言区加上命令`\usepackage{pst-coil}`就可以使用以下画螺旋形曲线的命令:

```
\pscoil[选项]{箭型}(起点坐标)(终点坐标)
\psCoil[选项]{起始角度}{终结角度}
\pszigzag[选项]{箭型}(起点坐标)(终点坐标)
```

其中`\pscoil`是画一个螺旋线; `\psCoil`画一个水平的螺旋线; `\pszigzag`画一个曲折线.

这些命令的选项有:

```
coilwidth=长度 指定宽度, 默认值1 cm;
coilheight=长度 指定长度, 默认值1;
coilarm=长度 螺旋的柄长, 默认值0.5 cm;
coilaspect=角度 方向角, 默认值45;
coilinc=角度 指定倾角, 默认值10.
```

例见12-6-9.tex的图107至110.

节点及其连线

节点名称应该是由字母与数字组成的字符串, 而且应该以字母开头. 命令中的位置是t, b, B, r, l, c的组合, 其含义参见244页`\rput`的定义, 默认值是居中. 建立节点的命令如下所示:

`\rnode[位置]{节点名称}{对象}` 建立一个具有指定名称的节点, 其形状为矩形. 选项位置确定了参考点的位置.

`\Rnode(x,y){节点名称}{对象}` 意义同`\rnode`, 只是它的参考点是盒子的中心加上 (x,y) .

`\pnode(x,y){节点名称}` 在 (x,y) 创建一个维数为0的节点, 默认位置是 $(0,0)$.

`\cnode[选项](x,y){半径}{节点名称}` 以指定半径画一个圆作为节点.

`\circlenode[选项]{节点名称}{对象}` 形状同`\pscirclebox`, 同时构造一个节点.

`\cnodeput[选项]{角度}(x,y){节点名称}{对象}` 把对象旋转角度并放在 (x,y) , 把它用圆围起来, 并构成一个节点.

`\ovalnode[选项]{节点名称}{对象}` 形状同`\psovalbox`, 同时构造一个节点.

创建了节点后往往需要用各种连线连接起来, 这些连线有以下选项:

`nodesep=长度` 设置节点外面的边界宽度, 默认值是0.

`offset=长度` 连线起点的偏移量, 默认值是0.

`angle=角度` 从节点出发连线的角度, 默认值是0.

`arcangle=角度` 仅用于`\ncarc`, 默认值是8.

`arm=长度` 与节点直接相连的分支(“手臂”)的长度, 默认值10 pt.

以下列举了连线的命令:

`\ncline[选项]{箭型}{节点A}{节点B}` 在节点间连一直线(可能有箭头). 例见12-6-10.tex的图111. 类似的命令是

`\pcline[选项]{箭型}(x_1,y_1)(x_2,y_2)`

`\nccurve[选项]{箭型}{节点A}{节点B}` 在节点间连一Bézier曲线. 见图112. 类似的命令是

`\pccurve[选项]{箭型}(x_1,y_1)(x_2,y_2)`

`\ncarc[选项]{箭型}{节点A}{节点B}` 是`\nccurve`的另一种形式. 见图113. 类似的命令是

`\pcarc[选项]{箭型}(x_1,y_1)(x_2,y_2)`

`\ncbar[选项]{箭型}{节点A}{节点B}` 用折线连接两个节点, 以角度`angleA`进入节点, 分支长度分别是`armA`与`armB`. 见图114. 类似的命令是

`\pcbar[选项]{箭型}(x_1,y_1)(x_2,y_2)`

`\ncdiag[选项]{箭型}{节点A}{节点B}` 用折线连接两个节点, 每个节点各伸出一个分支, 其长度与角度分别是`armA`, `armB`与`angleA`, `angleB`. 然后再用一条直线连接这两个分支的端点. 见图115. 类似的命令是

`\pcdiag[选项]{箭型}(x_1,y_1)(x_2,y_2)`

`\ncdiagg[选项]{箭型}{节点A}{节点B}` 与`\ncdiag`类似, 只是节点A没有分支伸出. 见图116.

`\ncangle[选项]{箭型}{节点A}{节点B}` 也用3条线段连接, 但是机器自动调节`armA`的长度, 使得从A发出的分支与中间线段相交成直角. 连接点由`angleA`与`angleB`确定, 且只能改变`armB`, 见图117. 类似的命令是

`\pcangle[选项]{箭型}(x_1,y_1)(x_2,y_2)`

`\ncangles[选项]{箭型}{节点A}{节点B}` 用4条线段连接, `armA`与`armB`都能改变, 从A出发的分支与相连的线段交成直角, 中间两条线段间也交成直角. 见图118.

`\ncloop[选项]{箭型}{节点A}{节点B}` 用5条线段连接, 因此A和B可以是同一个节点. 分支长由`arm`控制, 角度由参数`angle`控制, 先从A画一个分支, 然后

画 3 条互相垂直的线段, 使得第一条线段与由 A 伸出的分支垂直, 而第三条线段与由 B 伸出的分支相连, 其中第一条线段的长度等于 `loopsize`. 见图 119. 类似的命令是

`\pcloop[选项]{箭型}(x_1, y_1)(x_2, y_2)`

`\nccircle[选项]{箭型}{节点}{半径}` 画一个起终点都在同一点的圆, 见图 120.

在连线上注字

在连接线上注字的基本命令是

`\ncput[选项]{标注}`

`\naput[选项]{标注}`

`\nbput[选项]{标注}`

如果连线是自左向右的, 那么 `\ncput`、`\naput`、`\nbput` 分别把标注放在连线的中间、上面 (above)、下面 (below). 如果连线是自右向左的, 那就要把上下倒过来. 此外 `\ncput*` 会使连线中间断开放进标注. 见 12-6-10.tex 的图 121.

上述命令可以有以下选项:

ref=参照点位置 这里的参照点位置可以取 `t`, `b`, `B`, `r`, `l`, `c` 的组合, 其含义参见 244 页 `\rput` 的定义, 默认值是居中.

nrot=旋转角度 这里的旋转角度可以取 `\rput` 的选项中容许的值, 默认值为 0. 特别可取 `:U`, 表示与节点的连线平行. 见 12-6-10.tex 的图 122, 123.

npos=参数 这里的参数的值如果在 0 与 1 之间, 则标注位于第一个线段内; 如果在 1 与 2 之间, 则标注位于第二个线段内, 依此类推. 其默认值如下所示:

连线类型	线段数	范围	默认值
<code>\ncline</code>	1	$0 \leq \text{npos} \leq 1$	0.5
<code>\nccurve</code>	1	$0 \leq \text{npos} \leq 1$	0.5
<code>\ncarc</code>	1	$0 \leq \text{npos} \leq 1$	0.5
<code>\ncbar</code>	3	$0 \leq \text{npos} \leq 3$	1.5
<code>\ncdiag</code>	3	$0 \leq \text{npos} \leq 3$	1.5
<code>\ncdiagg</code>	2	$0 \leq \text{npos} \leq 2$	0.5
<code>\ncangle</code>	3	$0 \leq \text{npos} \leq 3$	1.5
<code>\ncangles</code>	4	$0 \leq \text{npos} \leq 4$	1.5
<code>\ncloop</code>	5	$0 \leq \text{npos} \leq 5$	2.5
<code>\nccircle</code>	1	$0 \leq \text{npos} \leq 1$	0.5

例见图 124.

pdfTricks

PSTricks 的最大缺点是与 pdf^LA_TE_X 不兼容. PdfTricks 就是为此设计的. 其作者是 Radhakrishnan、Rajagopal 和 A. Chambert-Loir. 不过这个包在与 MiKTeX

的兼容上有点问题, 因此 hooklee 作了改进, 命名为 `pdftricks_new`. 在使用时需在导言区加入 `\usepackage{pdftricks_new}`, 并且凡是 PSTricks 编译时需要用到的宏包都应该放在 `\begin{psinputs}...\end{psinputs}` 之内. 如果图形中出现汉字, 则也要把 CJK 包包括进去, 例如

```
\begin{psinputs}
  \usepackage{pstricks}
  \usepackage{pst-plot}
  \usepackage{pst-node}
  \usepackage{multido}
  \usepackage{CJK}
  .....
\end{psinputs}
```

在正文中用 `\begin{pdfpic}...\end{pdfpic}` 把 PSTricks 的绘图环境 `pspicture` 包裹起来. 如果图中出现汉字, 则要把 CJK 环境也包括在 `pdfpic` 环境之内. 如果在正文中也有汉字, 那么在导言区还要加入 `\usepackage{CJK}` 的命令. 例 12-6-11.tex 就是 12-6-10.tex 的 pdf 版, 用 pdfL^AT_EX 编译一次就能得到 pdf 文件.

还有一个有用的命令是

```
\NoProcess[用逗号与连字符分隔的数字序列]
```

表示方括号里面的图不需重新处理, 只要利用以前生成的 pdf 图形文件即可, 这是因为每一个 `pdfpic` 环境会生成一个 pdf 图像, 以 1, 2, ... 顺序排列. 例如 `\NoProcess[1,2,4-7]` 表示只需对第 3 个图重新生成 pdf 图形文件 (假定一共有 7 个图).

如想了解更多一些, 请看 `texmf/doc/latex/pdftricks/manual.pdf`.

§ 12.7 PGF

PGF (portable graphics format, 可携带图形格式) 是 beamer 包的作者 Till Tantau 开发的. 目的是建立一个类似于 PSTricks 的图形包, 既能应用于 dvips, 又能适用于 pdfL^AT_EX. 这样就能弥补 PSTricks 不与 pdfL^AT_EX 兼容的缺点. 当然它的功能还不及 PSTricks. 但是它目前处于不断开发的状态之下. 本节是根据用户指南编写的, 为了节省篇幅, 正文只介绍命令的用法, 把例子放在文件里, 请读者自己编译后观看. 英文原件请参看 `texmf/doc/latex/pgf/pgfuserguide.pdf`.

目前此宏包包含 `pgf.sty` (核心文件), `pgfarrows.sty` (画各种箭头), `pgfnodes.sty` (画节点与连线), `pgfshade.sty` (画阴影及渐变色彩). 使用时要在导言区用 `\usepackage` 命令读入相应文件.

pgf 绘图环境

```
\begin{pgfpicture}{左下角 x 值}{左下角 y 值}{右上角 x 值}{右上角 y 值}
.....
\end{pgfpicture}
```

这个命令建立了一个绘图环境。4 个坐标参数的意义是自明的。不过与 PSTricks 不同, 4 个坐标应该是带有单位的长度, 例如 3 mm, 5 pt 等。命令 `\pgfpictureboxed` 按指定的范围画一个方框, 对于调试十分有用。

在一个绘图环境内部可用

```
\begin{pgfscope}
.....
\end{pgfscope}
```

限制改变参数命令的作用范围。

如何指定一个点

`\pgforigin` 指坐标系的原点。例如: `\pgfmoveto{\pgforigin}`。

`\pgfpoint{x 坐标}{y 坐标}` 得到指定坐标的点, 坐标是 TeX 认识的长度。例: `\pgfline{\pgfpoint{10sp}{-1.5cm}}{\pgfpoint{10pt}{1cm}}`。

`\pgfpolar{幅角}{极半径}` 得到指定极坐标的点, 例如: `\pgfmoveto{\pgfpolar{30}{1cm}}`

`\pgfdirection{方向}` 方向的取值只能是: n, s, e, w, ne, nw, se, sw, 分别代表北、南、东、西、东北、西北、东南和西南。本命令给出这些方向所对应的角度。例如: `\pgfmoveto{\pgfpolar{\pgfdirection{n}}{1cm}}`。

`\pgfextractx{长度命令}{点}` 把指定点的 x 坐标赋予长度命令。例如:

`\newdimen\mydim`

`\pgfextractx{\mydim}{\pgfpoint{2cm}{4pt}}`

结果得到 `\mydim` 等于 2 cm。

`\pgfextracty{长度命令}{点}` 意义同上, 只是把 x 换成 y 。

`\pgfsetxvec{点}` 设置 x 单位向量, 此向量的默认值是 `\pgfsetxvec{\pgfpoint{1cm}{0cm}}`。类似的命令有 `\pgfsetyvec` 与 `\pgfsetzvec`。

`\pgfxy(x 坐标,y 坐标)` 定义一个点, 这里的坐标是一对数, 是相对于 x, y 单位向量的坐标。例如: `\pgfline{\pgfxy(0,0)}{\pgfxy(1,1)}`

`\pgfxyz(x 坐标,y 坐标,z 坐标)` 定义一个点, 这里的坐标是 3 个数, 是相对于 x, y, z 单位向量的坐标。例如 (见 12-7-1.tex 的图 1):

```
\begin{pgfpicture}{-0.385cm}{-0.385cm}{1cm}{1cm}
  \pgfline{\pgfxyz(0,0,0)}{\pgfxyz(0,0,1)}
  \pgfline{\pgfxyz(0,0,0)}{\pgfxyz(0,1,0)}
  \pgfline{\pgfxyz(0,0,0)}{\pgfxyz(1,0,0)}
\end{pgfpicture}
```

以下命令都把点看成以原点为始点的向量:

`\pgfdiff{点 p_1 }{点 p_2 }` 得到两个向量的差向量 $p_2 - p_1$. 例如:

`\pgfmoveto{\pgfdiff{\pgfxy(1,1)}{\pgfxy(2,3)}}`.

`\pgfrelative{点 p_1 }{点 p_2 }` 得到两个向量的和向量 $p_2 + p_1$. 例如:

`\pgfmoveto{\pgfrelative{\pgfxy(0,1)}{\pgfpoint{1pt}{2pt}}}`.

`\pgfpartway{比值 r }{点 p_1 }{点 p_2 }` 得到两点的定比分点 $p_1 + r(p_2 - p_1)$. 当比值 $r = 0.5$ 时得到中点. 例如:

`\pgfmoveto{\pgfpartway{0.5}{\pgfxy(1,1)}{\pgfxy(2,3)}}`.

`\pgfbackoff{距离}{始点}{终点}` 得到两点连线上相距始点等于距离的点. 例如:

`\pgfline{\pgfbackoff{2pt}{\pgfxy(1,1)}{\pgfxy(2,3)}}
{\pgfbackoff{3pt}{\pgfxy(2,3)}{\pgfxy(1,1)}}`

`\pgfcorner[方向]{第一个点}{第二个点}` 得到以此两个点为顶点的矩形在指定方向上的顶点. 矩形 4 个点的方向是 `se`, `ne`, `nw`, `sw`. 如果选取方向为 `s`, `n`, `e`, `w`, 则得到矩形边上的中点. 例见 12-7-1.tex 的图 2.

坐标系

可以对坐标系作平移、旋转或放缩, 但是这些运算与节点命令不相容! 注意: 在放缩时, 线条粗细也一起变化, 因此最好还是用改变单位向量的方法.

```
\begin{pgftranslate}{新原点}
.....
\end{pgftranslate}
```

把坐标系平移到新原点.

`\pgftranslateto{新原点}` 把坐标系平移到新原点.

`\pgfbox[水平位置, 竖直位置]{内容}` 这里的位置是指参考点在盒子中的方位, 水平位置的取值是 `left`, `right`, `center`, 竖直位置的取值是 `top`, `bottom`, `center`, `base`.

`\pgfputat{新原点}{命令}` 从新原点执行命令. 例如:

`\pgfputat{\pgfxy(1,0)}{\pgfbox[center,center]{你好}}.`

```
\begin{pgfrotateby}{旋转角的正切}
.....
\end{pgfrotateby}
```

对坐标系旋转指定的角度. 可以用命令 `\pgfdegree` 计算旋转角. 例如:

`\begin{pgfrotateby}{\pgfdegree{30}}.`

```
\begin{pgfmagnify}{x 放大倍数}{y 放大倍数}
.....
\end{pgfmagnify}
```

放大整个图形。

构造一个路径

这里介绍的一系列命令用于构造一个路径, 然后再用线条画出来或者用彩色填充. 为了构造一个路径, 首先要用 `\pgfmoveto` 移动到始点, 然后是一系列 `\pgflineto` 及 `\pgfcurveto` 命令. 你可以用 `\pgfclosepath` 创建一个闭路径.

`\pgfmoveto{点}` 以指定点作为当前点.

`\pgflineto{点}` 以当前点到指定点的直线段作为路径的扩充, 然后这个新点成为当前点. 例如: `\pgflineto{\pgfxy(1,1)}`.

`\pgfcurveto{支撑点1}{支撑点2}{点}` 以当前点到指定点的曲线段作为路径的扩充, 然后这个新点成为当前点. 当前点到支撑点1的直线以及支撑点2到指定点的直线都是这条曲线的切线. 例如 (参见 12-7-1.tex 的图3):

`\pgfcurveto{\pgfxy(1,1)}{\pgfxy(2,1)}{\pgfxy(2,0)}`

`\pgfclosepath` 连接当前点与路径的始点.

`\pgfzerocircle{半径}` 以原点为中心作一个圆加到当前路径.

`\pgfzeroellipse{轴向量1}{轴向量2}` 以原点为中心、按两个轴向量作一个椭圆加到当前路径, 参见 12-7-1.tex 的图4.

画线或填充

在构造好路径后, 用以下命令把路径画出来.

`\pgfstroke` 把路径画成线.

`\pgfclosetroke` 使当前路径闭合, 并画成线.

`\pgffill` 使路径闭合, 并用当前色彩填充.

`\pgfsetlinewidth{线条宽度}` 设定线条宽度. 参数值 0pt 是指设备能画出的最细宽度, 对于高分辨率的打印机, 这样的线条几乎看不出.

`\pgfsetdash{偶数个长度表}{位移值}` 使线条成为虚线. 长度表中的第一、三、五... 个值是实线的长度, 第二、四、六... 个值是空隙长度. 位移值是第一个实线段的起始位置. 例如: `\pgfsetdash{{0.5cm}{0.5cm}{0.1cm}{0.2cm}}{0cm}`, 参见 12-7-1.tex 的图5.

`\pgfsetbuttcap` 使以后的线条头部成平头, 这是默认的设置.

`\pgfsetroundcap` 使以后的线条头部成圆头.

`\pgfsetrectcap` 使以后的线条头部成突出的方头. 以上三条命令的不同效果可见下图, 其中最上面的是默认情形, 以后3个分别是平头、圆头与突出的方头. 参见 12-7-1.tex 的图6.



`\pgfsetbeveljoin` 使以后的线段连接处成斜角.

`\pgfsetroundjoin` 使以后的线段连接处成圆头.

`\pgfsetmiterjoin` 使以后的线段连接处成尖头. 以上三条命令的不同效果可见下图, 其中第一种是默认情形, 以后 3 个分别是斜角、圆头与尖头连接. 参见 12-7-1.tex 的图 7, 8.



裁剪

路径也可以用来以后的画面. 经裁剪后的图形不能再放大. 如果裁剪运算在 `pgfscope` 环境内部进行, 那么退出此环境后又会恢复原状.

`\pgfclip` 关闭当前路径, 并将这个路径与当前裁剪路径相交, 以得到新的裁剪路径. 参见 12-7-1.tex 的图 9.

`\pgfstrokeclip` 关闭并画出当前路径, 再将这个路径与当前裁剪路径相交, 以得到新的裁剪路径. 参见 12-7-1.tex 的图 10.

`\pgfclosestrokeclip` 封闭并画出当前路径, 再将这个路径与当前裁剪路径相交, 以得到新的裁剪路径. 参见 12-7-1.tex 的图 11.

`\pgffillclip` 封闭并填充当前路径, 再将这个路径与当前裁剪路径相交, 以得到新的裁剪路径.

`\pgffillstrokeclip` 封闭并填充当前路径, 且画出当前路径, 再将这个路径与当前裁剪路径相交, 以得到新的裁剪路径.

一些绘图命令

`\pgfline{始点}{终点}` 画一条直线段.

`\pgfxyline(x_1, y_1)(x_2, y_2)` 画一条直线段. 例如:

`\pgfxyline(0,0)(1,1).`

`\pgfcurve{始点}{支撑点 1}{支撑点 2}{终点}` 画一条曲线. 例如:

`\pgfcurve{\pgfxy(0,0)}{\pgfxy(0,1)}{\pgfxy(1,1)}{\pgfxy(1,0)}.`

`\pgfxycurve(x_1, y_1)(x'_1, y'_1)(x'_2, y'_2)(x_2, y_2)` 画一条曲线. 例如:

`\pgfxycurve(0,0)(0,1)(1,1)(1,0).`

`\pgfrect[类型]{左下角点}{矩形的宽度与高度}` 画一个矩形, 其中的类型可以是: `stroke`, `fill`, `fillstroke` 或 `clip`. 例如:

`\pgfrect[fill]{\pgfxy(2,2)}{\pgfxy(1,1)}.`

`\pgfcircle[类型]{圆心}{半径}` 画一个圆, 其中的类型可以是 `stroke`, `fill` 或 `fillstroke`. 例如:

`\pgfcircle[stroke]{\pgfxy(1,1)}{10pt}.`

`\pgfellipse[类型]{中心}{轴向量 1}{轴向量 2}` 画一个椭圆, 其中的类型可以是: `stroke`, `fill` 或 `fillstroke`. 例如:

`\pgfellipse[fill]{\pgforigin}{\pgfxy(2,0)}{\pgfxy(0,1)}.`

`\pgfgrid[选项]{左下角点}{右上角点}` 画坐标网格, 这里的选项可以是:

`stepx=长度` 设置水平步长, 默认值是 1 cm.

`stepy=长度` 设置竖直步长, 默认值是 1 cm.

`step=向量` 同时设置水平与竖直步长.

例见 12-7-1.tex 的图 12.

图像嵌入

pgf 包也有图像嵌入的功能, 而且图像文件可以不加扩展名, pgf 会根据情况自动添加扩展名. pgf 嵌入包含两步, 第一步是使用图像之前先用 `\pgfdeclareimage` 声明一个图像, 然后再使用命令 `\pgfuseimage` 嵌入图像. 这样, 即使这个图像在文中使用多次, 图像信息也只需读入一次, 这样可以大大缩小文件. 此外, pgf 还有一个命令 `\pgfimage` 可以同时完成以上两步, 立即嵌入文中.

`\pgfdeclareimage[选项]{图名}{文件名}` 声明一个图像, 但不画任何东西. 要画出一个图像, 需使用命令 `\pgfuseimage{图名}`. 文件名不带扩展名, 当输出 pdf 文件时, 会自动按 .pdf, .jpg, .png 顺序搜索; 当输出 ps 文件时, 会自动按 .eps, .epsi, .ps 顺序搜索.

可以使用的选项是:

`height=高度` 设置图像的高度. 如果不同时设定宽度的话, 将会保持图像的高宽比.

`width=宽度` 设置图像的宽度. 如果不同时设定高度的话, 将会保持图像的高宽比.

`page=页码` 指定多页文件中的页码. 这个选项有以下效果: pgf 首先寻找以下名字的文件:

文件名 . page 页码 . 扩展名

如果找到这样的文件, 就使用这个文件. 否则, pgf 使用名为 文件名 . 扩展名 的文件. 对于新版的 pdf^LA^TE_X, 只插入选定的页; 对旧版的 pdf^LA^TE_X 或 dvips, 则插入整个文件, 同时会发出一个警告信息.

`interpolate=true` 或 `false` 若为真, 则在放缩时会被“光滑化”, 默认是假.

`mask=蒙片名` 选取一个透明蒙片. 此蒙片必须用命令 `\pgfdeclaremask` (见后) 声明过的. 这个选项仅适用于 pdf, 不是所有浏览器都支持蒙片.

例如:

```
\pgfdeclareimage[height=1cm]{image1}{pgf-tu-logo}
```

`\pgfuseimage{图名}` 把以前声明过的图像插入正文. 如果你要在图形环境里使用此命令, 那么应该用 `\pgfbox` 把它括起来. 例如:

```
\pgfputat{\pgfxy(1,1)}{\pgfbox[left,base]{\pgfuseimage{image1}}}
```

参见 12-7-1.tex 的图 13.

如果宏 `\pgfalternameextension` 可以展开成一个非空的替换扩展名, 那么 pgf 将首先使用名为 图像名 . 替换扩展名 的文件, 如果这个图像未被定义, pgf 将在替换扩展名内查找字符 !, 如果找到了, 就把惊叹号连同它前面的字符从替换扩展名内删去, pgf 仍试图使用名为 图像名 . 替换扩展名 的文件, 如此继续, 直至替

换扩展名内不再包含惊叹号为止. 这时就使用原始图像. 可以使用以下命令定义 `\pgfalternatextension`:

```
\renewcommand{\pgfalternatextension}{!25!white}
```

`xxcolor`宏包里的 `colormixin` 环境会自动设置替换扩展名. 参见 12-7-1.tex 的图 14.

`\pgfaliasimage`{别名}{已有图名} 例如:

```
\pgfaliasimage{image.!30!white}{image.!25!white}
```

`\pgfimage`[选项]{文件名} 声明一个名为 `pgflastimage` 的图像, 并且立即使用它. 可以用 `\pgfaliasimage` 为 `pgflastimage` 取一个名字供以后再次使用. 参见 12-7-1.tex 的图 15.

`\pgfdeclaremask`[选项]{蒙片名}{文件名} 建立一个名为蒙片名的透明蒙片, 这个文件应该是只含灰度信息的图像, 图像尺寸必须与被蒙的图相同. 而且只能用于点阵式的图形, 如 .jpg 与 .png. 不能用于 .pdf 的图形. 参见 12-7-1.tex 的图 16 (只能用 pdf \LaTeX 编译才能正确显示).

为了加速编译, 可在导言区加入选项: `\usepackage[draft]{pgf}`, 这样在编译时就不读入图像, 只检验文件是否存在, 而且默认的图像高与宽都是 1cm.

注字

可以使用命令 `\pgfbox` 与 `\pgfputat` 来注字. 例如:

```
\pgfputat{\pgfxy(1,1)}{\pgfbox[left,base]{left}}
```

参见 12-7-1.tex 的图 17.

各种箭头

可以用以下命令规定线条的箭头形式:










`\pgfsetstartarrow`{箭型} 为以后的线条规定出发箭头的类型.

`\pgfsetendarrow`{箭型} 为以后的线条规定到达箭头的类型.

`\pgfclearstartarrow` 清除以前对出发箭头类型的规定.

`\pgfclearendarrow` 清除以前对到达箭头类型的规定.

箭型如下表所示 (参见 12-7-1.tex 的图 18):

	<code>\pgfarrowlargepointed{6pt}</code>
	<code>\pgfarrowtriangle{4pt}</code>
	<code>\pgfarrowcircle{4pt}</code>
	<code>\pgfarrowdiamond</code>
	<code>\pgfarrowdot</code>
	<code>\pgfarrowpointed</code>
	<code>\pgfarrowround</code>
	<code>\pgfarrowsquare</code>
	<code>\pgfarrowbar</code>

————→ `\pgfarrowsingle`

————→ `\pgfarrowto`

利用以下命令可以构造更复杂的箭型:

`\pgfarrowswap{箭型}` 使原来的箭头反向. 参见 12-7-1.tex 的图 19.

`\pgfarrowdouble{箭型}` 使原来的箭头重复一次. 参见 12-7-1.tex 的图 20.

`\pgfarrowtriple{箭型}` 使原来的箭头重复两次. 参见 12-7-1.tex 的图 21.

`\pgfarrowcombine{箭型 1}{箭型 2}` 画上两个箭头. 命令 `\pgfarrowcombine` 有相似的效果, 只是使两个箭头离得远一点. 参见 12-7-1.tex 的图 22.

在线上加标注

`\pgflabel{位置数}{始点}{终点}{间隙}` 在两点间连一条虚拟线, 间隙是指标注与直线的距离, 位置数规定了标注的位置, 始点是 0, 终点是 1, 中点是 0.5. 参见 12-7-1.tex 的图 23.

`\pgfputlabelrotated{位置数}{始点}{终点}{间隙}{绘图命令}` 这个命令使得绘图命令的坐标系原点移到指定的点上, 并旋转到虚拟直线上. 参见 12-7-1.tex 的图 24.

阴影

阴影要用到宏包 `pgfshade`. 只有新版的 `pdfLaTeX` 才能产生阴影效果. 而且 `gsview` 也可能显示效果不佳.

类似与一个图像, 阴影要先声明才能使用, 而且要放在一个 `TEX` 盒子里. 为了把阴影包含在 `pgfpicture` 内, 你要把它放在 `\pgfbox` 内.

有 3 种阴影: 水平、竖直及径向, 你还可以对阴影的片断作旋转, 以产生复杂的阴影. 一个阴影被声明后, 就能用命令 `\pgfuseshading` 使用它.

水平阴影就是一个水平条, 你至少要指定左右两端的色彩, 也可以在加上中间的色彩. 例如:

```
rgb(0cm)=(1,0,0); rgb(2cm)=(0,1,0); rgb(4cm)=(0,0,1)
```

意为左端 (0 cm) 是红色, 2 cm 处为绿色, 而 4 cm 处则为蓝色. 这里的位置应该按递增顺序给出.

`\pgfdeclarehorizontalshading[色彩表]{阴影名}{阴影高度}{色彩指示}`
声明一个水平阴影, 其长度由色彩指示的数据确定. 例如:

```
\pgfdeclarehorizontalshading{myshading}{1cm}{rgb(0cm)=(1,0,0);  
color(2cm)=(green); color(4cm)=(blue)}
```

参见 12-7-1.tex 的图 25.

色彩表作用是: 如果在声明阴影时使用色彩表里的色彩, 那么这个阴影要在使用时才根据当时的色彩表定义加以计算. 如果改变色彩表的定义, 阴影也相应改变. 而且在声明时并不要求色彩表已被定义. 参见 12-7-1.tex 的图 26.

`\pgfdeclareverticalshading[色彩表]{阴影名}{阴影宽度}{色彩指示}` 声明一个竖直阴影, 其含义同上一命令. 参见 12-7-1.tex 的图 27.

`\pgfdeclareradialshading[色彩表]{阴影名}{中心点}{色彩指示}` 声明一个径向阴影, 其半径由色彩指示的数据确定. 如果中心点就是原点, 那么原点的色彩就是 0cm 的色彩, 圆的半径是色彩指示中的最大长度. 如命令的中心点不是原点, 那么阴影圆的大小及位置都与上一情形相同, 只是命令中心点的色彩是 0cm 的色彩, 而其他地方的色彩经变形后填满这个圆. 参见 12-7-1.tex 的图 28.

`\pgfuseshading{阴影名}` 插入已经声明过的阴影. 如用在 pgfpicture 环境内则应放在 `\pgfbox` 内. 参见 12-7-1.tex 的图 28.

`\pgfaliasshading{别名}{已有阴影名}` 例如:

`\pgfaliasshading{shading!30}{shading!25}`

创建节点

使用节点功能要用到宏包 pgfnodes. 使用节点的好处之一是当你需要改变节点位置时, 所有的连线都会跟着改变, 不需一一修改.

在下列命令中, 绘图类型可以是: `stroke`, `fill`, `fillstroke` 或 `virtual` (什么也不画).

`\pgfnodecircle{节点名}[绘图类型]{中心}{半径}` 创建一个圆形节点. 参见 12-7-2.tex 的图 25.

`\pgfnodeirect{节点名}[绘图类型]{中心}{宽度与高度向量}` 创建一个矩形节点. 参见 12-7-3.tex 的图 26.

`\pgfnodebox{节点名}[绘图类型]{中心}{TeX 文本}{水平间隙}{竖直间隙}` 创建一个矩形节点, TeX 文本放在其中, 与边框保持给定的间隙. 参见 12-7-4.tex 的图 27.

节点的相对坐标

`\pgfnodecenter{节点名}` 以当前位置为节点中心, 这个命令尤其适用于建立相对于其他节点的节点. 参见 12-7-2.tex 的图 28.

`\pgfnodeborder{节点名}{角度}{边界间隙}` 先确定此节点边界上位于指定方向 (以度为单位的角度) 上的点, 当边界间隙取正值时, 这个点再外移指定距离, 负值则内移. 参见 12-7-2.tex 的图 29.

`\pgfconnstart[边界间隙]{始节点}{终节点}` 先确定两节点连线与边界的交点, 然后在连线上移动边界间隙指定的距离. 参见 12-7-2.tex 的图 30.

节点的连线

`\pgfnodesetsepstart{偏移量}`, `\pgfnodesetsepend{偏移量}` 设置连线开始部分与结束部分的偏移量. 参见 12-7-2.tex 的图 31.

`\pgfnodeconncurve{始节点}{终节点}` 从第一个节点的边界到第二个节点的边界画一条连线.

`\pgfnodeconncurve{始节点}{终节点}{起始角度}{终结角度}{ d_1 }{ d_2 }` 画一条曲线连接两个节点, 连线在节点边界的取指定的角度, d_1, d_2 分别是支撑点到边界的距离. 参见 12-7-2.tex 的图 32.

在连线上加标注

`\pgfnodelabel{始节点}{终节点}{位置数}[竖直间隙]{绘图命令}` 在两节点连线上按位置数确定的点. 参见 12-7-2.tex 的图 33.

`\pgfnodelabelrotated{始节点}{终节点}{位置数}[竖直间隙]{绘图命令}` 同上面的命令, 只是把标注旋转成平行于连线. 参见 12-7-2.tex 的图 34.

彩色包的扩展

本节介绍与 pgf 一起分发的宏包 `xxcolor`, 它是 `xcolor` 的扩展, 后者又是 `color` 的扩展. `xxcolor` 的主要功能是提供一个环境, 使得其中的颜色变淡褪色.

```
\begin{colormixin}{混色指示}
.....
\end{colormixin}
```

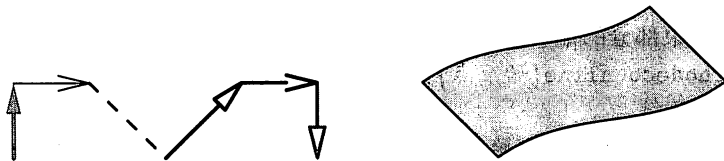
混色指示的形式是数!色彩, 其中的数不超过 100. 例如: `90!blue` 意为用 10% 的蓝色与 90% 的原有色彩混合. `25!white` 使得原有色彩几乎褪尽. 参见 12-7-2.tex 的图 35.

请注意这个环境只能改变标准 L^AT_EX 的 `\color` 命令设定的色彩, 而且不会改变图像的色彩. 但是这个环境会把 `\pgfalternateextension` 设置成混色指示, 例如按图 35 被设置成 `50!white`. 而深入到下一层嵌套环境后, 又变成 `50!white!25!black`. `\pgfuseimage` 会根据预定的规则寻找相应的图像文件.

`\colorcurrentmixin` 展开成当前的混色指示. 参见 12-7-2.tex 的图 36.

§ 12.8 其他绘图宏包

另一个配合 TeX 的绘图宏包是 Peter Kabal 开发的 T_EXdraw. 这也是一个自由软件. T_EXdraw 的输出是含有 PostScript 命令的附加文件, 生成 dvi 文件后要用 dvips 转化成 ps 文件才能在 PostScript 打印机上打印出图像. 不过现在有不少 dvi 驱动软件也能打印 T_EXdraw 作出的图像. 由于 PostScript 的强大功能被调动起来, 因此 T_EXdraw 的特色是产生各种形状以及灰度的箭头图, 还能在闭合图形内加阴影. 不过它的曲线图形的功能不多, 除了 Bézier 曲线外, 只能画圆弧和椭圆弧. 下附的两个图显示了 T_EXdraw 的特色, 左图的源程序如下 (参见 12-8-1.tex).



```
\begin{texdraw}
\drawdim cm
\linewidth 0.06 \setgray 0.6 \arrowheadtype t:F
\avec(0 1)
\linewidth 0.02 \setgray 0 \arrowheadtype t:V
\avec(1 1)
\linewidth 0.03 \lpatt(0.15 0.2) \lvec (2 0)
\linewidth 0.04 \lpatt() \arrowheadtype t:T
\avec(3 1)
\arrowheadtype t:H \avec(4 1)
\setgray 0.4 \arrowheadtype t:W \avec(4 0)
\end{texdraw}
```

第十三章 输出到投影仪或互联网

本章介绍 L^AT_EX 如何扩展它的领域, 从生成纸质文本发展到供屏幕阅读、投影仪演示或网上浏览的多种格式文件. 根据我们的经验, 本章选择几个有代表性的宏包加以介绍. 在 13.2 节介绍如何用 pdfL^AT_EX 生成供电脑阅读的 PDF 文件, 第 13.3 节介绍生成用于屏幕上阅读或者演示的 pdfscreen 包. 这个包的好处是已有不少现成模板可供套用. 13.4 节介绍的是最新也是功能最强的生成演示文稿的 beamer 包, 如果肯花功夫仔细阅读它的用户手册的话, 一定能做出不亚于 PowerPoint 的演示文稿. 对于心急的读者来说, 只要选取一个模板, 运用自己的 L^AT_EX 知识, 马上就能做出一个漂亮的文稿. 值得大家一试. 可是 beamer 与 PSTricks 不兼容, 喜欢用 PSTricks 的读者也可选用 prosper, 不过本书不介绍了. 13.5 节介绍的 SliTeX 是和 L^AT_EX 同龄的宏包, 主要用于生成供光学投影仪使用的黑白透明片, 还有它的存在价值. 最后两节介绍了两个从 L^AT_EX 源文件生成 HTML 网页文件的程序 L^AT_EX2HTML 以及 L^AT_EX4ht, 它们转换中文原文件时输出的仍是汉字, 对于生成中文网页特别合适. 用于生成网络数学教材或辅导材料有它无可替代的优点.

§ 13.1 如何显示彩色

为了在屏幕上显示彩色或打印彩色页面, 先要用

```
\usepackage[可选项]{xcolor}
```

输入彩色宏包. 其中的可选项有几类. 第一类是彩色驱动程序的名称, 可用的有: dvips, xdvi, dvipdf, dvipdfm, dvipdfmx, pdftex, dvipsone, dviwindo, emtex, dviwin, oztex, textures, pctexps, pctexwin, pctexhp, pctex32, truetex, vtex, tcidvi, xetex. 一般总是用 dvips. 第二类是彩色模式, 可用的有: natural, rgb, cmv, cmk, hsb, gray, RGB, HTML, HSB, Gray, monochrome. 默认值是 natural, 也就是说尊重源文件中定义色彩所选取的模式. 如果选取其他参数, 例如 cmk, 那么 L^AT_EX 在编译时就会自动把源文件中所有的色彩自动转换成 cmk 模式输出. 这主要用于有特殊要求的彩色打印机或显示器, 否则你可以什么也不选. 最后一个选项 monochrome 是取消彩色输出. 第三类选项是装入预定义的色彩集合, 可用的有: dvipsnames, dvipsnames*, svgnames, svgnames*, x11names, x11names*. 建议选取 dvipsnames, 与参数 dvips 配合, 提供了 68 种预定义的色彩.

在所有的彩色驱动程序中都被预定义的色彩名有: red, green, blue, yellow, cyan, magenta, black, white. 此外, 在 xcolor 包内还补充定义了 9 个色彩: orange, violet, purple, brown, pink, olive, darkgray, gray, lightgray. 这 17 种色彩可以用作生成其他色彩的基础. 如果加入选项 dvipsnames, 就使得能使用的预定义色彩达到 68 种. 例 13-1-1.tex 给出了 dvipsnames 预定义的 68 种色彩.

定义色彩的命令为

```
\definecolor{色彩名}{模式}{数据}
\providecolor{色彩名}{模式}{数据}
```

它们给色彩名指定一种颜色, 在以后的命令里可以使用这种色彩名. 例如 `\definecolor{red}{rgb}{1,0,0}`. 常用的模式有: rgb (红, 绿, 蓝), cmyk (青, 洋红, 黄, 黑), gray (灰) 等. 数据是一组用逗号分隔的 0 与 1 之间的十进小数串, 表示每个分量的强度. 例如 `[rgb]{1,0,0}` 定义了红色, `[cmyk]{0,0,1,0}` 定义了黄色, 模式 gray 只取一个参数. 这两条命令的区别是: 如果色彩名已经有定义, 那么 `\providecolor` 不起作用, 而 `\definecolor` 则覆盖原有的定义.

使用色彩的命令有

```
\color{色彩}
\textcolor{色彩}{文本}
\pagecolor{色彩}
```

其中 `\color` 命令表示以后的文本就用这种颜色打印, 它的作用范围持续到所在环境的结束或者遇到另一条 `\color` 命令为止. 命令 `\textcolor` 是用指定的色彩打印文本. 命令 `\pagecolor` 设定从本页开始的背景色, 以后的页都使用此背景色, 直至再遇到 `\pagecolor` 命令为止.

这些命令中的色彩可以是已经定义的色彩名, 也可以是不同色彩的混合. 我们不想详细介绍色彩混合的语法, 只是从实用的观点举几个例子, 读者看了后就知道怎样用了. 例如 `\color{ANamedColor!75}` 表示取一种已定义为 ANamedColor 的色彩的 75% 的色调; `\color{green!40!yellow}` 表示 40% 绿色与 60% 黄色的混合色; `\color{-green!40!yellow}` 则是上述色彩的补色;

`\color{rgb:-green!40!yellow,3;green!40!yellow,2;red,1}`
则是 3 份第一种色彩加上 2 份第二种色彩再加上 1 份第三种色彩的混合.

以下是彩色盒子的命令:

```
\colorbox{色彩}{文本}
```

生成一个以文本为内容的 LR 盒子, 并以指定的色彩作为盒子的背景色.

```
\fcolorbox{色彩一}{色彩二}{文本}
```

生成一个以文本为内容的带框 LR 盒子, 以色彩一作为框线颜色, 色彩二作为背景色.

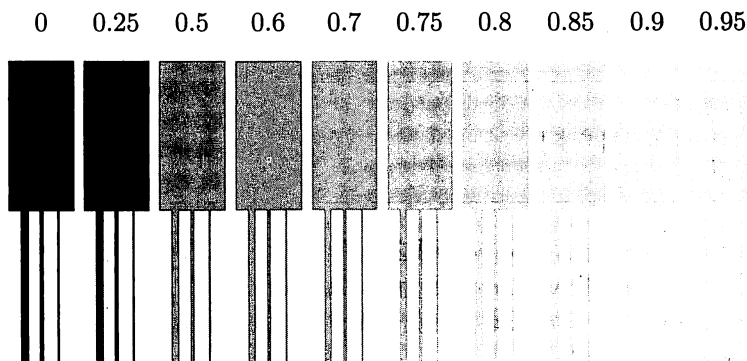
```
\normalcolor
```

把文字的颜色重置为前言末尾所激活的颜色, 通常是黑色. 前言中的 `\color` 命令可以改变‘标准’色彩.

关于 `xcolor` 包的更多内容, 请参看 `texmf\doc\latex\xcolor\xcolor.pdf`.

比较好的工作方式是把要使用的色彩都用 `\definecolor` 定义好, 正文中只使用已定义的色彩名, 这样可以根据打印结果再作调整. 请注意同一种色彩使用不同的打印机会得到很不相同的效果, 屏幕显示与打印结果也有很大差距, 因此微调是必不可少的. 请读者试试编译及显示例 13-1-2.tex, 你会发现在预览器 yap 里不显示背景色, 要经 `dvips` 转换成 `ps` 文件后才能得到正确显示.

下面的表格显示了不同灰度等级的打印效果. 第一行数字表示 `gray` 模式下的参数. 例如 0 就是 `black`, 0.25 是 `darkgray`, 0.5 是 `gray`, 0.75 是 `lightgray`, 1 就是 `white`.



§ 13.2 如何使用 pdf_TE_X

使用 `dvipdfm` 可以直接把 `dvi` 文件转换成 `pdf` 文件, 不过它不会增加 `LATEX` 中没有的 `pdf` 的功能. 因此了解 `pdfTEX` 的功能与用法是有好处的.

`pdfTEX` 可用于处理 Plain `TEX` 以及 `AMS-TEX` 文件, 而 `pdfLATEX` 则可直接处理 `LATEX` 文件. 一般的 `LATEX` 源文件经 `pdfLATEX` 处理后就得到了相应的 `pdf` 文件, 但是带彩色的或利用 `graphics` 宏包嵌入图形的源文件必须加以修改, 因为 `pdfLATEX` 不能直接处理 `eps` 格式的图形文件.

利用本书光盘中所含的安装程序安装的 WinEdt 集成环境都已经在工具条上建立了 `pdfLATEX` 按钮, 用鼠标点击即可.

使用彩色与嵌入图形

如果要使用彩色与嵌入图形, 则应选取 `pdftex` 作为彩色驱动程序, 而且应改用宏包 `graphicx`, 即在前言中声明

```
\usepackage[pdftex]{graphicx,xcolor}
```

这样就能重新得到例 13-1-1 的彩色表 (参见 13-2-1.tex 以及 13-2-2.tex).

pdf^TE_X 中能使用的图形格式有: pdf, png, tif(tiff), jpg(jpeg), 但不支持 eps, 为了在 pdf^TE_X 中使用 eps 图形, 可先调用 epstopdf.exe 把它转换成 pdf 格式.

graphicx 宏包的嵌入命令是:

```
\includegraphics[键= 值, ...]{图形文件名}
```

有些键的值是整数, 有的键值则为逻辑值, 即 true 或 false, 对这样的键, 单列键名而不输入键值相当于输入真值. 键与键之间用逗号分开. 可用的键有 (注意有些键仅对 eps 图形有效):

scale= 数: 放大倍数;

width= 长度: 输出图形的宽度, 如果 height 不给出, 则按同一比例放缩;

height= 长度: 输出图形的高度, 如果 width 不给出, 则按同一比例放缩;

totalheight= 长度: 输出图形的总高度 (高度加深度), 如果要旋转图形, 则应该用它取代 height;

keepaspectratio (=true/false): 如果同时指定了 width 与 height, 那么这个标志保证原始图形的高宽之比不变, 同时又使变换后的图形不超出指定的宽与高;

angle= 数: 作逆时针旋转的角度 (以度为单位), 它与键 width 与 height 的出现顺序会影响最终得到的图形 (参见例 13-2-3.tex);

origin= 位置: 规定旋转的中心, 默认值是 bl, 表示左下角, 也可取: c (中心), t (顶部), r (右端), B (基线), 以及它们的组合;

draft (=true/false): 不插入图形, 仅保留出图形所占位置, 显示图形边界方框及图形文件名. 这可明显加快处理速度;

clip (=true/false): 截去边框外面的图形 (仅适用于 eps 图形);

bb= 左下角横坐标 左下角纵坐标 右上角横坐标 右上角纵坐标: 规定图形边框的坐标, 4 个长度间用空格隔开, 可以带有单位, 默认单位是大点 (bp) (仅适用于 eps 图形);

viewport= 左下角横坐标 左下角纵坐标 右上角横坐标 右上角纵坐标: 以图形的左下角作为坐标原点, 指定图形的边框. 与 clip 连用可截取图形的一个部分 (仅适用于 eps 图形);

trim= 左下角横坐标差值 左下角纵坐标差值 右上角横坐标差值 右上角纵坐标差值: 指定图形向内截去的值 (仅适用于 eps 图形);

hiresbb (=true/false): 在插入的 eps 图形文件中不使用通常的 %BoundingBox 的值, 而是用 %HiresBoundingBox 的值作为图边界盒子的值 (仅适用于 eps 图形).

graphicx 宏包的旋转盒子的命令是

```
\rotatebox[键= 值, ...]{角度}{文本}
```

可使用的键 origin 同上所示, 另一种指定旋转中心的方式是: $x = \text{长度}$, $y = \text{长度}$.

上述 graphicx 宏包在 pdfL^AT_EX 中的应用示例可见 13-2-3.tex.

对文字或图形作变换

还可以利用 pdf 的原始命令对图形作仿射变换, 这需要使用 pdfT_EX 的特有命令, 这些命令都是以 \pdf 起头的. 如下面的变换命令

```
\pdfliteral{q a b c d e f cm}%
...
\pdfliteral{Q}
```

其中 a, b, c, d, e, f 是 6 个实数, 它们刻画了一个仿射变换, 其长度单位是 bp:

$$\begin{cases} x' = ax + cy + e \\ y' = bx + dy + f \end{cases} \quad \text{或} \quad (x' \ y') = (x \ y) \begin{pmatrix} a & b \\ c & d \end{pmatrix} + (e \ f)$$

这里的 'q', 'cm', 'Q' 都是 pdf 的命令. 因此逆时针旋转 t° 的变换可描述成 $a = \cos t$, $b = \sin t$, $c = -\sin t$, $d = \cos t$, $e = f = 0$ (三角函数值应化成十进小数输入). 此外被变换的图形或文字应该放在一个长宽高都被设成 0 的盒子里, 在有关的命令之间不能有空格插入, 因此每条命令的后面要用注解号 '%' 除去后面的空格. 为了使后继的内容有正确的留空, 必须作手工调整. 如下面的例子 (参见 13-2-3.tex):

```
\newsavebox{\testbox}
\sbox{\testbox}{\Huge 变换后的文字.}
\ht\testbox=0pt \wd\testbox=0pt \dp\testbox=0pt
这是\pdfliteral{q 0.866 -0.5 0.5 0.866 0 0 cm}%
\usebox{\testbox}%
\pdfliteral{Q}
```

也可以用

```
\sbox{\testbox}{\includegraphics{pic.jpg}}
```

形式的命令对图形作变换.

其他图形文件

Xy-pic 宏包可以直接在 pdfL^AT_EX 下运行, T_EXdraw 则不行. MetaPost 生成的文件可用以下命令输入 (在前言中要有 \usepackage[pdftex]{graphicx}):

```
\convertMPtoPDF{文件名}{横向放大倍数}{纵向放大倍数}
```

这里的文件名必须是运行 mpost 后生成的. 例如 13-2-3.tex 中的命令

```
\convertMPtoPDF{fig.1}{1.5}{0.9}.
```

插入注记

用 \pdfannot 命令可以建立注记, 下面是一个实例 (参见例 13-2-4.tex):


```
\pdfannot width 10cm height 0cm depth 4cm
{
  /Subtype /Text      % 文本性质的注解
  % /Open true        % 如果取消注解号 '%', 则注解窗口是打开的
  /Contents(This is ...) % 注解内容, 其中不能有汉字
}
```

其中的尺寸是可选的. 也可插入声音或动画文件作为注记(例13-2-4.tex):

```
\pdfannot width 0in height 0in depth 0in
{
  /Subtype /Sound
  /Sound << /F (SoundFile.wav) >>}
```

```
\pdfannot width 4in height 0in depth 3in
{
  /Subtype /Movie
  /Movie << /F (MovieFile.mov) >>}
```

声音文件也能作为动画(Movie)播放.

建立链接

以下命令可建立链接的目标, 请注意 pdf 的链接是指向一个页, 不再深入到具体的位置.

```
\pdfdest 目标说明
```

目标说明的第一部分是指定一个用作标识的正整数或名字, 格式是 num 数字或 name{标识}; 第二部分是目标页显示的格式, 其选项如下表所示:

fit	窗口里显示整页
fith	使页宽适应窗口
fitv	使页高适应窗口
xyz	到当前位置, 后面可接 zoom 放大因子, 放大因子的值是整数, 相当于真实因子的 1000 倍

例如(参见 13-2-4.tex):

```
\pdfdest name{dest1} xyz zoom 1500
```

建立一个链接可使用以下命令:

```
\pdfstartlink [尺寸] [属性] 动作说明
.....
\pdfendlink
```

其中的可选项尺寸具有 `width ... height ... depth ...` 的形式, 其默认值是包含在 `\pdfstartlink` 与 `\pdfendlink` 之间的内容的范围, 可以超过一页; 可选项属性说明边框的颜色与厚度, 例如

```
attr{/C [0.9 0 0] /Border [0 0 2]}
```

规定边框为暗红色; 动作可以是 `goto` 或 `user`, 如果是 `goto`, 后面应指明 `num` 数字, `name{标识}` 或 `page 页码 {/Fit}`, 如果想链接到另一个 pdf 文件, 可使用 `goto file{文件名}`; 用户定义的动作可参见下例:

```
user{/Subtype /Link
/A << /Type /Action
/S /URI
/URI (http://www.tug.org/) >>}
```

下面是链接的一个例子 (参见 13-2-4.tex):

```
本链接指向
\pdfstartlink height 15pt depth 5pt
attr{/C [0.9 0 0] /Border [0 0 2]}
goto name{dest1}
name\{dest1\}
\pdfendlink
```

建立书签

为了建立 pdf 文件的中文书签, 可在导言里加上以下声明, 调入宏包 `hyperref`:

```
\usepackage[CJKbookmarks]{hyperref}
```

不过生成书签需要编译二次. 如果不使用中文书签, 当然可以略去 `CJKbookmarks` 这个选项. 参见 13-2-3.tex. 如果显示的书签仍为乱码, 这是因为 Acrobat Reader 只能正确处理 pdf 文件中使用 Unicode 编码的书签, 而 CJK 排版生成的是 GBK 编码, 因而生成的 pdf 文件可能在书签中出现乱码. 为此 hooklee 在 `cxtex` 编制的程序基础上改进成 `gbk2uni.exe`, 此文件存放在 `texmf\cct\bin` 里. 当执行两遍 `pdflatex` 后, 如果发现书签是乱码, 就可执行以下命令:

`gbk2uni` 不带后缀的输入文件名

再执行一遍 `pdflatex`, 就能显示中文的书签了! (这部分内容摘自张林波的《关于新版 CCT 的说明》, 欲知详情, 可见随书光盘的 `Doc\常用中文文档\关于新版 CCT 的说明 (张林波 20041220).pdf`. 由于 hooklee 在制作随书光盘时已经把 `gbk2uni.exe` 的处理加进去了, 因此只要安装正确, 自然会生成中文书签.

更详细的用法说明可参看 `texmf\doc\pdftex\pdftex-a.pdf`.

§ 13.3 pdfscreen 与 pdfslide

13.3.1 用 pdfscreen 生成适合屏幕阅读的 pdf 文件

pdfscreen 是 C. V. Radhakrishnan 开发的, 它利用 pdfTeX 生成适合于屏幕上阅读的材料, 同时又能按正常的大小用 dvi 打印输出, 这一切都由选项 print 或 screen 控制. 在源文件里可以使用

```
\begin{print}...\end{print}
或
\begin{screen}...\end{screen}
```

形式的命令把只用于打印或屏幕显示的内容包裹起来.

需要在导言区加入读宏包命令:

```
\usepackage[screen,panelleft]{pdfscreen}
```

而且这个命令最好放在导言的末尾, 以防止其他命令改变它的设置. pdfscreen 会自动输入宏包 graphicx 与 color. 方括号里的可选项有以下几种:

screen	产生适合屏幕输出的版本
print	产生适合打印输出的版本, 与 dvi 一样
panelleft	导航板居左
panelright	导航板居右
nopanel	取消导航板
paneltoc	把目录放在面板上, 但只要正文中出现 \tableofcontents, 本命令即失效.
sectionbreak	在每一节的前面分页

导航面板与按钮有 6 种配色方案可供使用: bluelace (浅蓝), blue (蓝色), gray (灰色), orange (桔黄), palegreen (浅绿) 与 chocolate (浅棕色), 默认的是蓝色.

以下命令应出现在导言中:

\screensize{高度}{宽度} 规定屏幕尺寸, 用户可以自由取值, 但必须出现.

\margins{左边}{右边}{顶部}{底部} 规定页边宽度, 必须出现.

\overlay{图形文件} 选取背景图形.

\overlayempty 无背景图形.

\paneloverlay{图形文件} 选取导航板的背景图形.

\paneloverlayempty 导航板无背景图形.

\changeoverlay 出现本声明后, 每出现一个新节, 它的背景图形都会自动改变, 也就是改变颜色, 变化范围从 overlay0.pdf 至 overlay10.pdf, 共 11 种, 循环使用.

`\urlid{URL 地址}` 导航板上 ‘Home Page’ 按钮指向的 URL 地址.

导航板上的按钮名称有多种语言可供选择, 可惜没有中文, 因此我们只能在导言里将它们重新定义, 具体方法请参看例 13-3-1.tex.

定义新的按钮可使用以下命令:

```
\addButton{按钮宽度}{名称}
\imageButton{宽度}{高度}{图形文件名}
```

后者是以图形作为按钮. 但是为使按钮发生作用, 必须把它包含在某个动作里. 例如:

`\Acrobatmenu{动作}{\addButton 或 \imageButton...}` 点击此按钮就会启动一个 Acrobat 动作, 这里的动作可以是: `FirstPage` (跳转第一页)、`PrevPage` (跳转前一页)、`NextPage` (跳转下一页)、`LastPage` (跳转最后一页)、`GoBack` (退回)、`GoToPage` (跳转指定的页)、`FullScreen` (全屏显示)、`Close` (关闭本文件)、`Quit` (退出 Acrobat).

`\href{http://网址}{\addButton 或 \imageButton...}` 链接到指定的网址.

`\hyperlink{标识}{\addButton 或 \imageButton...}` 跳转到标识处, 标识可用 `\pdfdest named` 定义.

还有 4 个命令:

```
\bottombuttons      \nobottombuttons
\topbuttons         \notopbuttons
```

它们用来控制一个放在页底或页顶的小导航板, 依次表示设置或不设置页底导航板, 以及设置或不设置页顶导航板, 页底与页顶可以同时设置, 并不冲突. 它们的作用从出现命令的这一页开始, 一直延续到遇到相反的命令为止, 也可出现在导言中.

详细用法说明可参见随书光盘里的 Doc\常用中文文档\pdfscreen 中文手册 (汤银才编译 20040423). 英文原件见 `texmf\doc\latex\pdfscreen>manual-screen.pdf`. 已有 TeX 爱好者利用 pdfscreen 制作了论文模板, 请参见随书光盘的 Doc\中文TeX模板\幻灯模板\[pdfscreen模板(游道德之平林)].

13.3.2 用 pdfslide 生成演示用 pdf 文件

pdfslide 也是 C. V. Radhakrishnan 开发的, 它利用 pdfTeX 生成适合外接投影仪使用的文件. 可是根据作者的试验, 在 pdfslide 环境里的 `\section` 命令与 CJK 不完全兼容, 主要是 `\section` 命令与汉字有冲突, 解决方法是使用 `\section*`, 即可避开这个问题, 参见例 13-3-2.tex.

为了适应投影仪的特点, pdfslide 对字体作了放大, 它使用的 `\normalsize` 的字体大小相当于 16pt, 而希望使用 ‘正常’ 大小的用户也可使用

冠以 `\real...` 的系列命令, 例如: `\realnormalsize` (相当于 12pt), `\reallarge` (相当于 14pt) 等.

`pdfslide` 的另一个特点是可以利用 Acrobat 的页面过渡效果, 利用命令

```
\pagedissolve{选项}
```

可以产生各种过渡效果. 下表列出了一些可以使用的效果:

效果名	解 释
Split	两条线扫过屏幕展开新页, 类似于剧场的开幕或闭幕.
Blinds	类似于 Split, 但是有多条线同时进行, 好像打开百叶窗.
Box	通过从中心开始不断扩大的矩形展示新页.
Wipe	单独一根线扫过页面展开新页.
Dissolve	旧页‘溶化’, 产生新页.
Glitter	类似于 Dissolve, 不过效果从一边扫到另一边.
R	简单地以新页取代旧页, 无特殊效果, 这是默认值.

对某些效果还可以加上参数, 如下表所示:

键	解 释
/D	效果持续时间, 单位是秒 (适用于所有效果).
/Di 角度	指示 Wipe 或 Glitter 的运动方向, 角度应是 90 的倍数, 按逆时针方向计算, 0 表示自左向右.
/Dm	可能的值为 /H 或 /V, 分别表示水平或垂直的效果 (适用于 Split 及 Blinds).
/M	/O 表示效果从中心向边缘扩展, /I 则与之相反 (适用于 Split 及 Box).

例如:

```
\pagedissolve{Wipe /D 1 /Di 90}
\pagedissolve{Glitter /D 3 /Di 180}
\pagedissolve{Split /D 2 /Dm /H /M /O}
```

更详细的用法说明可参看 `texmf\doc\pdfslide>manual.pdf`.

§ 13.4 用 beamer 生成演示用 pdf 文件

Beamer 是 Till Tantau 在 2003 年创立的, 后经不断完善, 成了目前用于演示的最佳宏包之一. 本节的内容主要取自 Tantau 写的用户手册, 全文可见 200 多页的 `texmf\doc\latex\beamer\beameruserguide.pdf`.

Beamer 的主要特点有

- 可以同时适用于 pdf_latex 以及 latex+dvips;
- 可以逐段显示, 也可以显示动态效果;
- 预先设计好的各种主题样式给你很大的选择余地;
- 可以通过改变整体参数以重新设置版面、色彩、字体大小等, 同时你仍可保持对细节的完全控制;
- 很容易实现讲稿与演示材料间的互相转换;
- 最终输出的 PDF 文件能保证到处适用, 而且显示效果不受机器系统的影响.

13.4.1 快速入门

现在我们通过 beamer 说明书里的一个搞笑实例来说明如何使用 beamer 建立演示文稿. 这个例子的英文版是 13-4-1en.tex, 中文版是 13-4-1.tex. 读者在这个例子的基础上很容易做出自己的讲演文稿. 由于英文版与中文版有同样多的需求, 我们先介绍一个英文的实例, 然后说明如何稍作修改得到中文版.

话说某作者 Euclid 被邀请在一个学术会议上作讲演. 他就选了一个模板 texmf\doc\latex\beamer\solutions\conference-talks\conference-ornate-20min.en.tex, 将它改名另存在自己的目录里. 然后他打开此文件, 在 \title 后面的花括号里填入报告的标题. 如果标题太长, 可以在中间的方括号里填入一个简短的标题, 让这个短标题显示在每个幻灯片上. 然后再在 \author, \date 后面填上作者姓名和会议名称. 如有必要, 也可以用 \subtitle 命令加上副题, 用 \institute 注明作者的单位. 这些信息填写完成后, 用 \begin{document} 命令进入正文.

在 beamer 里, 每一帧都包含在一个 frame 环境里:

```
\begin{frame}
  \frametitle{...}
  \framesubtitle{...}
  .....
\end{frame}
```

一个帧可以包含覆盖、逐段显示等内容, 所以由若干幅幻灯片组合而成. 其中 \frametitle 规定了每帧的标题, \framesubtitle 则规定了副标题. 标题帧和目录帧分别用以下命令输入

```

\begin{frame}
  \titlepage
\end{frame}
\begin{frame}
  \frametitle{Outline}
  \tableofcontents
\end{frame}

```

如果想要使目录逐节显示, 只要在目录后加一个参数即可

```
\tableofcontents[pausessections]
```

在组织演示文稿时, 可以先用 `\section`、`\subsection` 命令建立节与小节. 它们的标题除了显示在目录里外, 也会显示在每帧的上方以及 Acrobat Reader 的书签栏里. 一般的讲演不宜超过 3 节, 每节至多包含 3 小节.

在输入演示内容时, 可以利用预定义的环境 `theorem`, `lemma`, `corollary`, `proof`, `definition` 以及 `example` 等. 这些环境都有定义好的彩色方框突出显示. 与此类似的是 `block` 环境:

```

\begin{block}{标题}
  .....
\end{block}

```

它的输出类似于定理环境, 但是可以自选标题. 此外, 命令 `\alert` 可以用来突出显示一些词语. 通过编译实例 13-4-1en.tex 可以看到这些命令的效果.

Beamer 还有一个 `columns` 环境把一些内容分栏并列, 在环境内部通过命令 `\column{宽度}` 建立新栏, 例如

```

\begin{columns}
  \column{.5\textwidth}
  .....
  \column{.5\textwidth}
  .....
\end{columns}

```

这些栏在竖直方向上是居中对齐的, 如果你想要使它们顶端或底部对齐, 只要加上选项 `[t]` 或 `[b]` 即可, 如 `\begin{columns}[t]`. 例 13-4-1en.tex 中就有这样的实例, 不过被注解掉了, 只要把注解号 “%” 去掉, 就可以编译显示了.

根据演示的特点, 应该多使用罗列环境 `itemize`, `enumerate` 等. 在这些环境里插入 `\pause` 命令, 可使各个条目逐条显示. 例如:

```

\begin{itemize}
  \item 2 is prime (two divisors: 1 and 2).
  \pause
  \item 3 is prime (two divisors: 1 and 3).
  \pause
  \item 4 is not prime (\alert{three} divisors: 1, 2, and 4).
\end{itemize}

```

也可以使用命令:

```

\beamerdefaultoverlayspecification{<+>}

```

使得在此命令以后上述罗列环境以及其他一些环境自动实现逐段显示.

如果想对逐段显示作更精细的控制,就需要形如<-3,5-6,8->的控制命令,表示这段内容的显示范围是:第3幅以前、第4至6幅以及第6幅以后.例如:

```

\begin{frame}
\begin{enumerate}
  \item<1->Suppose  $p$  were the largest prime number.
  \item<2->Let  $q$  be the product of the first  $p$  numbers.
  \item<3->Then  $q + 1$  is not divisible by any of them.
  \item<1->Thus  $q + 1$  is also prime and greater than  $p$ .
\end{enumerate}
\uncover<4->{The proof used \textit{reductio ad absurdum}.}
\end{frame}

```

其中接在\item后面的<1->、<2->、<3->规定了本项内容应该从第几幅开始显示.而\uncover<4->命令则规定其后花括号里的内容应该从第4幅开始显示.与之类似的命令是\only<4->,同样使后面花括号从第4幅开始显示.这两条命令不同之处在于:\uncover与其他覆盖的命令一样,被其覆盖的内容在不被显示时仍然占据位置,而\only所控制的内容则在不显示时完全消失,因此使用\only命令会使版面位置发生变化.

抄录环境verbatim仍然适用于beamer,不过要加上fragile选项.但是beamer提供了一个更强的环境semiverbatim,它类似于verbatim,只是\, {与}继续作为控制符使用,要打印这些字符必须输入\\, \{与\}.例如:


```

\begin{frame}[fragile]
\frametitle{An Algorithm For Finding Primes Numbers.}
\begin{semiverbatim}
\uncover<1->{\alert<0>{int main (void)}}
\uncover<1->{\alert<0>{\{\}}}
\uncover<1->{\alert<1>{ \alert<4>{std::}vector<bool> is_prime
(100, true);}}
\uncover<1->{\alert<1>{ for (int i = 2; i <100; i++)}}
\uncover<2->{\alert<2>{ if (is_prime[i])}}
\uncover<2->{\alert<0>{ \{\}}}
\uncover<3->{\alert<3>{ \alert<4>{std::}cout <<i <<" ";}}
\uncover<3->{\alert<3>{ for (int j = i; j <100;}}
\uncover<3->{\alert<3>{ is_prime [j] = false, j+=i);}}
\uncover<2->{\alert<0>{ \{\}}}
\uncover<1->{\alert<0>{ return 0;}}
\uncover<1->{\alert<0>{\{\}}}
\end{semiverbatim}
\visible<4->{Note the use of \alert{\texttt{std::}}.}
\end{frame}

```

请读者仔细分析一下这段输入, 设想其输出该是按怎样的规律逐段显示的, 然后再编译并显示例 13-4-1en.tex, 看看是否与你预测的一致. 这里新出现的命令 \visible 的意义基本相当于 \uncover, 其差异在于: 如果使用了命令 \setbeamercovered{transparent} 使得被覆盖的内容变得透明 (依稀可见), \visible 仍然使被覆盖的内容不可见.

就这样, Euclid 完成了他的演示材料, 得到了源文件 13-4-1en.tex, 请读者用 pdflatex 处理两遍 (或更多), 就能在 Acrobat Reader 里欣赏你的成果了.

现在我们再看看如何准备中文材料. 在存放本书例题的目录里有一个名为 beamer 模板的子目录, 里面有 3 个文件, 读者可以根据需要选取其中之一建立你自己的模板. 为了使用汉字, 要给 beamer 加上选项 [cjk], 并读入宏包 CJK. 如果你需要用到 theorem, definition 等环境, 则有必要加入“定理”、“定义”等汉字, 为此在导言区就要用 CJK 环境加注汉字. 又因在导言区处理 \title 等命令似乎有问题, 所以把它们移到正文部分, 除了抄录环境不能用中文外, 其他没有特别之处. 请读者编译并显示 13-4-1.tex 以观察效果. 这些例子也能作为你的模板.

如果你想得到 PS 文件, 只要用 L^AT_EX 处理两遍 (或更多), 然后用命令

```
dvips -P pdf 13-4-1en.dvi
```

处理即可. 其中选项 -P pdf 是告诉 dvips 使用 Type 1 轮廓字库. 如果发现这样的输出有问题, 则可以取消此选项, 改用通常的 Type 3 点阵字库. 不过你可能会发现这样得到的图像只有 128 mm×96 mm 大. 为了放大到一张 A4 纸大小, 可以执行以下命令:

```
dvips -P pdf -ta4 13-4-1en.dvi -o 13-4-1en.temp.ps
```

```
psnup -1 -W128mm -H96mm -pa4 13-4-1en.temp.ps 13-4-1en.ps
```

为了得到合适的页边留空, 应该在 `psnup` 的命令中插入选项 `-m1cm`, 表示留空 1 cm. 如果你想在一张 A4 纸上打印 2、4 或 6 幅画面, 可把 `psnup` 的第一个选项 `-1` 改为 `-2`、`-4` 或 `-6`. 此外为了使画面互相分离, 应再插入选项 `-b1cm`.

不过用这样得到的 PS 文件打印出来的硬拷贝也许并不是你所需要的. 因为它包含了所有的动态画面, 如果你使用了覆盖, 那么每一幅幻灯片都有一个画面, 像一幅幅动画一样, 显然太繁琐了. 你需要的是供分发的打印稿, 为此可在引入 `beamer` 类时加入选项 `[handout]`, 重新编译后再打印, 就能得到合意的硬拷贝了. 不过当你的幻灯片采用白色背景时, 打印出来会看不到边框. 为此只要在导言区加入以下命令:

```
\mode<handout>{\beamertemplatesolidbackgroundcolor{black!5}}
```

就会在使用 `handout` 选项输出时采用浅灰色作背景, 而供演示使用的背景不变. 此外, 你还可能会需要打印一份透明片, 以备机器故障时使用, 为此你可以把 `beamer` 的选项取为 `[trans]`, 再用同法打印在透明片上.

同样的内容也可以用论文的形式打印, 为此只要用以下两行命令:




```
\documentclass{article}
\usepackage{beamerarticle}
```


取代 `\documentclass{beamer}`. 参见


`texmf/doc/latex/beamer/examples/beamerexample2.article.tex`.


导航图标


用 Acrobat Reader 浏览时, 可用 `Ctrl-L` 键控制全屏与窗口显示间的切换. 在全屏显示时按下 `Esc` 键也会恢复到窗口显示. 这里还要对出现在幻灯片右下方的导航图标作一介绍. 图标共分 9 类:


- ◀  ▶ 是控制单幅幻灯片 (slide) 的图标, 点击左边的箭头会跳转到上一幅幻灯片, 点击右边的箭头会跳转到下一幅幻灯片, 点击中间的方框则会弹出一个窗口, 可让你跳转到任何一幅幻灯片.
- ◀  ▶ 是控制帧 (frame) 的图标, 点击左边的箭头会跳转到上一帧的最后一幅幻灯片, 点击右边的箭头会跳转到下一帧的第一幅幻灯片. 点击中间图标的左端会跳转到当前帧的第一幅幻灯片, 点击其右端则会跳转到当前帧的最后一幅幻灯片 (跳过中间的覆盖).
- ◀  ▶ 是控制小节 (subsection) 的图标, 点击左边的箭头会跳转到上一小节的最后一幅幻灯片, 点击右边的箭头会跳转到下一小节的第一幅幻灯片. 点击中间图标的左端会跳转到当前小节的第一幅幻灯片, 点击其右端则会跳转到当前小节的最后一幅幻灯片.


 是控制节 (section) 的图标, 点击左边的箭头会跳转到上一节的最后一幅幻灯片, 点击右边的箭头会跳转到下一节的第一幅幻灯片. 点击中间图标的左端会跳转到当前节的第一幅幻灯片, 点击其右端则会跳转到当前节的最后一幅幻灯片.

 是控制演示 (presentation) 的图标, 点击此图标的左端会跳转到整个演示的第一幅幻灯片, 点击其右端则会跳转到整个演示 (不包括附录) 的最后一幅幻灯片.

 是控制附录 (appendix) 的图标, 点击此图标的左端会跳转到整个附录的第一幅幻灯片, 点击其右端则会跳转到整个附录的最后一幅幻灯片. 如果没有附录就不显示此图标.

 是后退 (back) 图标, 点击此图标会跳转到刚才显示的幻灯片 (需要浏览器支持此功能).

 是前进 (forward) 图标, 点击此图标会跳转到已显示过的下一幅幻灯片 (需要浏览器支持此功能).

 是搜寻 (find) 图标, 点击此图标弹出一个搜寻窗口 (需要浏览器支持此功能).

读者可以用 Acrobat Reader 打开文件 `texmf\doc\latex\beamer\examples\beamerexample1.pdf` 以试验各导航图标的功能, 为得到好的效果, 最好是用全屏显示.

对于一个简单的演示不需要那么多的导航图标, 我们可以使用命令:

```
\setbeamertemplate{navigation symbols}{\insert...navigation symbol}
```

定制所需的导航图标. 这里的 ... 可以是 `slide`, `frame`, `subsection`, `section`, `doc` (演示与附录图标), `back`, `find`, `forward` (后退、搜寻与前进), 根据需要选择, 并列的 `\insert...` 命令之间用逗号分隔. 如果第二个花括号内没有任何内容, 导航图标就完全消失. 例如可以用

```
\setbeamertemplate{navigation symbols}{\insertslidenavigation symbol,
\insertdocnavigation symbol}
```

使得导航图标只出现 3 种.

这样我们已经快速学习了 beamer 的用法, 在我们的例子或模板的基础上, 不难写出你自己的演示文稿. 再次提醒一下: 中文模板放在本书练习当前目录的子目录 `beamer` 模板里. 英文模板可在 `texmf\doc\latex\beamer` 的子目录 `solutions` 里找到. Beamer 的例子放在子目录 `examples` 里. 建议读者把子目录里的文件拷贝出来编译一下试试. 不过例 `beamerexample1.tex` 需要修改, 给 beamer 类加上选项 `[table]`, 否则会出错. 例 `beamerexample4.tex` 也需要修改, 把 CJK 环境的参数 `{GB}` 改成 `{GBK}`.

13.4.2 高级技巧

帧的设计

一个演示可分解成一系列的帧(frame), 一个帧可能包含多幅幻灯片(slide). 创立一个帧的一般命令如下:

```
\begin{frame}<覆盖指示>[<默认覆盖指示>][选项]
...内容...
\end{frame}
```

其中\begin{frame}这一行的其他内容都是可以省略的.

第一个参数<覆盖指示>规定这个帧的哪几幅幻灯片能被显示. 例如<beamer>规定仅在beamer环境里才被显示; <1-2,4->规定除第3幅外均被显示; <0>表示完全不显示, <handout:0>规定在handout版本里不显示这个帧.

第二个参数[<默认覆盖指示>]的内容仅适用于当前帧, 常用的是[<+>]. 设置了这个参数后, 当前帧的内容被自动分段显示.

如果没有设置[<+>], 在帧的内容里也不含覆盖指示, 那么所有的内容都会显示在同一幅幻灯片里. 如果你希望拆分成几幅幻灯片显示, 可以设置选项[allowframebreaks, allowdisplaybreaks]. 帧的标题会重复出现在每幅幻灯片里, 后面加上罗马数字注明第几幅. 如果略微超长, 又不想拆开, 可以使用选项[shrink=20], 表示可以压缩不超过20%. 为了美观, 数字20可以改成5, 10, 一个一个试过去.

另一个可用的选项是[plain]. 设置此选项后, 幻灯片的页眉页底以及导航条等都被取消, 只剩下帧的内容. 这主要用于显示特大的图像或自己设计标题页.

每幅幻灯片的整体构图是由你选择的“主题”确定的. Beamer类已经预定义了各种主题供用户选择, 请读者参见13.4.3.

覆盖的技巧

使用覆盖的最简单方法是在一条命令后面加上<覆盖指示>. 例如命令\textbf<2-4>{ABC}使得ABC在第2、3、4幅幻灯片里显示成黑体, 而在其他情形则保持原先的字体. 类似的命令有\textit, \textsl, \textrm, \textsf, \color, \alert, \structure等. 如果命令带有其他参数, 则应放在覆盖指示之后, 如\color<2-3>[rgb]{1,0,0}. 此外, <覆盖指示>也可用于约束整个环境, 只要将它放在\begin{环境}之后即可. 例如:

```
\begin{theorem}<2->...\end{theorem}
```

\onslide是一个很特别的命令. 单用\onslide会使以后的内容始终被显示, 直到遇到其他覆盖命令为止, 而且它的作用范围不受TeX群组的约束, 也就是说, 它能“穿透”TeX命令. \onslide<覆盖指示>相当于\uncover<覆盖指示>. 但是\onslide命令具有穿透力, 与\uncover不同. \onslide+<覆盖指示>相当于\visible<覆盖指示>, 也就是后面的内容在覆盖指示的范围以外是完全不可见的. 同样地, \onslide+命令具有穿透力, 与\visible不同.

但是当 `\onslide` 后面接有用花括号括起来的文本时, 它的意义如下:

`\onslide<覆盖指示>{文本}` 相当于 `\uncover<覆盖指示>{文本}`;

`\onslide+<覆盖指示>{文本}` 相当于 `\visible<覆盖指示>{文本}`;

`\onslide*<覆盖指示>{文本}` 相当于 `\only<覆盖指示>{文本}`. 复习一下: 在不显示时, `\only` 命令作用范围的文本会完全消失, 不占位置.

有时我们会需要这样的效果: 第一幅里出现的文字或图画在第二幅里被新的文字或图画所替代. 这时可使用以下形式的命令:

```
\includegraphics<1>{first.pdf}
\includegraphics<2>{second.pdf}
\includegraphics<3>{third.pdf}
```

或

```
\only<1>{第一句.}
\only<2>{第二句.}
\only<3>{第三句.}
```

但是当三段话的高度不同时, 就会使周围的文本上下跳动. 为此, beamer 定义了一个新环境 `overprint`:

```
\begin{overprint}[宽度]
... 内容 ...
\end{overprint}
```

这个环境占用的高度相当于其中内容的最大高度, 把这块区域保留作为替换之用, 这样就不会影响此环境以外的文本的位置了.

下面我们给出一些使用覆盖的小技巧. 我们把这些例子收录在 `13-4-2.tex` 内, 供读者自己编译后观看效果.

在罗列环境里突出显示当前项:

```
\begin{itemize}
\item<1-| alert@1> First point.
\item<2-| alert@2> Second point.
\item<3-| alert@3> Third point.
\end{itemize}
```

或

```
\begin{itemize}[<+ -| alert@+>]
\item First point.
\item Second point.
\item Third point.
\end{itemize}
```

其中 `\alert@1` 的意义是用 `\alert` 作用在第 1 幅幻灯片上, `\alert@+` 表示增量作用, 即按 1, 2, ... 的规则作用, `<+>` 是同样的意义.

逐行显示带编号的公式

假定你有分成 3 行的公式:

```
\begin{align}
A &= B \\
&= C \\
&= D
\end{align}
```

如果使用命令 `\pause` 或 `\onslide` 会出错. 用以下方式处理是可以的:

```
\begin{align}
A &= B \\
&\uncover<2->{&= C} \\
&\uncover<3->{&= D}
\end{align}
```

但是你发现第 3 个公式的编号始终被显示. 因此这只能用于不对公式编号的 `align*` 环境. 问题是在于最后一行的编号出现在所有的幻灯片里. 因此我们在最后加一个不编号的行, 然后再用 `\vspace` 命令把基线移回来.

```
\begin{align}
A &= B \\
&\uncover<2->{&= C} \\
&\uncover<3->{&= D} \\
&\notag
\end{align}
\vspace{-1.5em}
```

逐行显示表格

当表格中有竖直线和水平线时, 逐行显示表格就会遇到问题. 因为在读进 `\onslide` 命令之前, 最左边的竖线就已经画好了. 因此尽量不要在表格里划线, 代替办法是用 `\rowcolors` 命令给表格的行着上不同的颜色. `\rowcolors` 是 `xcolor` 包的命令, 为了使用它需要在引入 `beamer` 类时加上选项 `[table]`, 其用法为

```
\rowcolors[命令]{行号}{奇数行色彩}{偶数行色彩}
```

此命令必须放在表格环境的外部, 其中的命令将在表格的每一行之前执行, 此命令通常为 `\hline` 或 `\noalign{...}`. 行号是指从这一行开始执行颜色命令, 然后按奇数行与偶数行交错着色. 其中 `blue!20` 表示 20% 的蓝色. 至于环境 `tabular` 的列格式中的字符 `!`, `<` 都是 `colortbl` 包定义, 用于引出后面的动作.

```

\rowcolors[] {1}{blue!20}{blue!10}
\begin{tabular}{l!{\vrule}cccc}
  Class & A & B & C & D \\ \hline
  X      & 1 & 2 & 3 & 4 \\ \pause
  Y      & 3 & 4 & 5 & 6 \\ \pause
  Z      & 5 & 6 & 7 & 8
\end{tabular}

```

逐列显示表格

此时把覆盖指示放在 tabular 环境的头部, 注意最后一列的 \onslide 不加参数, 这是为了保证第一列能够正确显示. 在这种情形里使用水平直线是很困难的.

```

\rowcolors[] {1}{blue!20}{blue!10}
\begin{tabular}{l!{\vrule}c<{\onslide<2->}c<{\onslide<3->}
c<{\onslide<4->}c<{\onslide}c}
  Class & A & B & C & D \\
  X      & 1 & 2 & 3 & 4 \\
  Y      & 3 & 4 & 5 & 6 \\
  Z      & 5 & 6 & 7 & 8
\end{tabular}

```

改变罗列条目前面的标记

有时你需要罗列一些待解决的问题, 并在前面打上叉, 然后一个一个讨论, 同时把叉改成勾. 叉与勾可在包 pifont 内找到. 解决这个问题的方法是建立一个名为 ballot 的新环境, 当这个环境的条目被激活时, 前面的标记同时被改变. 建立新环境的程序如下, 里面的命令看不懂也没有关系.

```

\newenvironment{ballotenv}
{
\only{%
  \setbeamertemplate{itemize item}{\ding{56}}%
  \setbeamertemplate{itemize subitem}{\small\ding{56}}%
  \setbeamertemplate{itemize subsubitem}{\footnotesize
    \ding{56}}}}
{}
\setbeamertemplate{itemize item}{\ding{52}}
\setbeamertemplate{itemize subitem}{\small\ding{52}}
\setbeamertemplate{itemize subsubitem}{\footnotesize\ding{52}}

```

下面是应用这个环境的例子:

```

\begin{itemize}
  \item<1-| ballot@1> First point.
  \item<2-| ballot@2> Second point.

```

```
\item<3-| ballot@3> Third point.
\end{itemize}
```

与

```
\begin{itemize}[<+-| ballot@+>]
  \item First point.
  \item Second point.
  \item Third point.
\end{itemize}
```

目录、参考文献、附录

显示目录的命令是

```
\tableofcontents[用逗号分隔的选项]
```

其中比较有用的选项有

`currentsection` 或 `currentsubsection` 仅对当前节 (相应地: 当前小节) 内的目录正常显示, 其他的内容均用半透明字体显示.

`hideallsubsections` 不显示所有的小节目录.

`hideothersubsections` 不显示当前节以外所有的小节目录.

`pausesections` 使目录按节逐段显示, 相当于在每节的目录后加上 `\pause` 命令.

Beamer 的参考文献录仍然沿用 L^AT_EX 的环境 `\begin{thebibliography}` ... `\end{thebibliography}`, 环境内部用 `\bibitem` 引入每条文献. 不过这个环境应该放在一个帧 (frame) 之内, 当超过一帧时应该另起一帧, 其中再建立一个新的 `thebibliography` 环境. 此外, 每个用 `\bibitem` 引导的条目中不同性质的内容 (如作者、书名、期刊名等) 应该用 `\newblock` 分隔, 如下例所示:

```
\bibitem[Dijkstra, 1982]{Dijkstra1982}
  E.~Dijkstra.
  \newblock Smoothsort, an alternative for sorting in situ.
  \newblock {\em Science of Computer Programming}, 1(3):223--233,
  1982.
```

beamer 会根据你所选定的主题, 对于用 `\newblock` 分隔的不同部分采取不同的显示方式, 例如分行显示或者用不同颜色, 以示区分.

命令 `\appendix` 的后面都是附录部分. 附录部分的节与小节都不出现在目录里. 附录应该是通常不被显示的内容, 只是在回答提问或需要深入说明时才会用到的内容.

添加超链接与按钮

建立超链接目标的命令是


```
\hypertarget<覆盖指示>{目标标签}{文本}
```

当这条命令当前的帧含有多幅幻灯片时, 覆盖指示不可缺少, 而且必须是唯一的幻灯片. 命令中的文本只出现在被指定的幻灯片中, 而在其他片中完全不出现.

超链接的命令是

```
\hyperlink<覆盖指示>{目标标签}{文本或按钮}
```

命令内的文本或按钮会在覆盖指示的范围内正常显示, 只要点击这个部位就会跳转到目标标签所在的幻灯片. 这些文本或按钮在覆盖指示的范围外完全消失. 这里的覆盖指示并不是必须的.

上述超链接命令可以被以下命令所代替, 这些命令已经包含了明确的跳转目标:

```
\hyperlinkslideprev<覆盖指示>{文本或按钮}  跳转到前一幅幻灯片;  
\hyperlinkslidenext<覆盖指示>{文本或按钮}  跳转到后一幅幻灯片;  
\hyperlinkframestart<覆盖指示>{文本或按钮}  跳转到当前帧的第一幅幻灯片;  
\hyperlinkframeend<覆盖指示>{文本或按钮}  跳转到当前帧的最后一幅幻灯片;
```

```
\hyperlinkframestartnext<覆盖指示>{文本或按钮}  跳转到下一帧的第一幅幻灯片;  
\hyperlinkframeendprev<覆盖指示>{文本或按钮}  跳转到上一帧的最后一幅幻灯片.
```

上列后四条命令中的 `frame` 可以改成 `subsection` 或 `section`, 也就是把帧换成相应的小节或节.

```
\hyperlinkpresentationstart<覆盖指示>{文本或按钮}  跳转到整个演示的第一幅幻灯片;  
\hyperlinkpresentationend<覆盖指示>{文本或按钮}  跳转到整个演示的最后一幅幻灯片.
```

上列两条命令中的 `presentation` 可以改成 `appendix` 或 `document`, 也就是把演示换成相应的附录或文件.

定义按钮的命令是

```
\beamerbutton{按钮文本}
```

这些按钮用在 `\hyperlink` 命令里就能产生超链接的作用. 类似的命令还有

```
\beamergotobutton{按钮文本}  生成一个带有小箭头的按钮;  
\beamskipbutton{按钮文本}  生成一个带有双重小箭头的按钮;  
\beamerreturnbutton{按钮文本}  生成一个带有指向左方的小箭头的按钮.
```

给环境加选项

`enumerate` 环境的默认计数形式是 1. 2. 3. 但 `\begin{enumerate}[选项]` 可以改变其形式. 这里的选项可以是 `[(1)]`, `[(i)]`, `[A.]`, ... 等等. 也就是在方括号里给出第一项的编号样式, 以后就会自动依此生成.

对于 description 环境, 可以用 `\begin{description}` [最长文本] 的形式设定此环境的缩进量.

对于 block 环境, 有以下命令

```
\begin{block}<覆盖指示>{块标题}
```

类似的环境有 alertblock 以及 exampleblock 等.

对于 theorem 以及 example 环境, 可以用 `\begin{theorem}` [补充文本] 的形式加上定理的补充信息, 例如 [Author, 2005]. 而 proof 环境有所不同, 命令 `\begin{proof}` [证明的名称] 内方括号的内容是取代默认的 Proof. 的. 例如可以用 [证明] 把这个环境汉化.

插入多媒体素材

为插入多媒体素材并使其能在 Acrobat Reader 里播放, 需要使用 multimedia.sty, 这是 beamer 包内的一个文件, 但不会自动读入. 故在导言区应该加上

```
\usepackage{multimedia}
```

对电影或声音素材, 可用以下命令插入:

```
\movie[选项]{布告文本}{音像文件名}
```

注意: 在最终生成的 PDF 文件里并不包含此音像文件, 因此这个文件必须与生成的 PDF 文件放在同一个目录里. 如果在 [选项] 里没有规定方框的大小, 就会以布告文本占据的方框作为输出电影的方框. 因此布告文本一般取为一幅图像, 它与放映的电影有相同的大小, 在电影未放映前, 听众看到的就是这个图像. 插入音像的命令的例子如下:

```
\movie{\includegraphics{myposterimage}}{mymovie.avi}
\movie[width=3cm,height=2cm,poster]{}{mymovie.mpg}
```

`\movie` 命令的选项有

autostart 在此幻灯片显示的同时开始播放音像, 一旦幻灯片换页, 音像也同时停止;

borderwidth=宽度 指定边框宽度, 当小于 0.5 pt 时可能无法显示;

depth=长度, height=长度, width=长度 指定放映框的深度、高度与宽度;

label=音像标签 供 `\hyperlinkmovie` 用的标签;

poster 用电影的第一幅画面作为布告. 一般来说, 当电影未被播放时是没有画面的;

repeat 反复播放;

showcontrols 显示控制条.

这段音像可以作为超链接的目标, 其命令如下:

```
\hyperlinkmovie[选项]{音像标签}{文本}
```

不过被链接的音像标签必须在同一页. 这里的选项与前面提到的相同, 但是有更高的优先级.

另外有一个专用于播放声音的命令:

```
\sound[选项]{布告文本}{声音文件名}
```

如果使用选项 `inlinesound` 就可以把声音文件的内容嵌入生成的 PDF 文件, 否则也要把用到的声音文件放在同一目录里. 为了保证播放, 声音文件应该是未经压缩的 wav 文件. 而且这一命令仅适用于 `pdflatex`, 对 `dvips` 无效. 这条命令的好处是可以同时播放几个声音, 不像 `\movie` 一次只能打开一个. 这条命令有类似于 `\movie` 的选项, 因此我们下面只列举几个特有的选项:

`automute` 在离开本页时, 停止所有声音;

`mixsound` 与正在播放的其他声音混放, 默认情形是停止其他声音单播当前声音.

如果要播放背景声, 可使用命令 `\sound[autostart]{\}{applauss.wav}`.

切换

切换命令用于规定当前的幻灯片如何替代前一幅, 因此它是属于当前页的. 不论出现在一页的什么地方, 都具有相同的效果. 切换是通过 Acrobat Reader 实现的, 只在全屏显示时才起作用. 这些切换命令的一般形式为

```
\命令<覆盖指示>[选项]
```

命令后面的参数都是可以省略的. 选项命令有两种:

`duration=秒数` 默认值是 1, 往往取 0.2 更合适;

`direction=度数` 允许值是 0, 90, 180, 270, 对于 `glitter`, 也可取 315.

切换命令有 (略去可选参数):

`\transblindshorizontal`, `\transblindsvertical` 用多条线扫过屏幕展开新页, 前一条命令使用水平线, 后一条用竖直线.

`\transboxin` 用一个矩形由外向中心收缩展开新页.

`\transboxout` 用一个矩形由中心向外展开新页.

`\transdissolve` 旧页溶化, 产生新页.

`\transglitter` 类似于 `glitter`, 不过从一边到另一边.

`\transsplithorizontalin`, `\transsplitverticalin` 两条线由外向内扫过屏幕展开新页, 类似于剧场闭幕. 前一条使用水平线, 后一条用竖直线.

`\transsplithorizontalout`, `\transsplitverticalout` 两条线由中心向外扫过屏幕展开新页, 类似于剧场开幕. 前一条使用水平线, 后一条用竖直线.

`\transwipe` 单独一根线扫过页面产生新页.

13.4.3 整体设计——主题

为了方便用户, beamer 已经设计了一系列演示主题, 它们规定了版面、色彩、字体等等要素. 对于一般用户来说, 只要用以下命令选择一个主题即可

```
\usetheme[选项]{主题名}
```

除了少数例外, beamer 的主题都是以作者们去过的城市命名的, 可被分为 5 大类, 我们将一一介绍其特点. 读者只要打开 beameruserguide.pdf, 从第 134 页开始可以看到各种主题的外形, 从而选择适合这次讲演的主题. 当然也可以改变几种主题分别编译比较, 从中选择一种最满意的.

无导航条的主题: 其中 default 是最简朴的, 也是默认的主题. boxes 可供用户自己设计顶部或底部的盒子. Bergen 的左边有一竖条, 但是要调整好有一些难度, 需要研究用户手册的有关内容. Boadilla 和 Madrid 可以在顶部插入当前节名与小节名. 此外还有 Pittsburgh 与 Rochester.

有树形导航条的主题包括 Antibes, JuanLesPins 与 Montpellier 3 种主题.

有目录竖条的主题包括 Berkeley, PaloAlto, Goettingen, Marburg 与 Hannover 5 种主题. 这些主题适宜于冗长的讲演, 因为它们含有目录条.

有圆点导航条的主题包括 Berlin, Dresden, Darmstadt, Frankfurt, Singapore 与 Szeged 6 种主题. 导航条里有一条由圆点组成的进度表.

有节与小节导航条的主题包括 Copenhagen, Luebeck, Malmos 与 Warsaw 4 种主题.

§ 13.5 用 slides 制作透明片

L^AT_EX 的 slides 类是专门用于制作投影片的, 这是因为投影片有它的特点, 它的字体应该大些, 矮胖些, 而且不在段落中间换页, 并且往往需要彩色. 因此 slides 类使用无衬线字体, 而且 \normalsize 字体相当于 20 pt. 例如导言部分可以输入

```
\documentclass{slides}
\usepackage[dvips]{xcolor}
.....
\begin{document}
.....
\end{document}
```

slides 里有 3 种环境: slide, overlay, note. 它们分别用以下方式进入:

```
\begin{slide} 文本及命令 \end{slide}
\begin{overlay} 文本及命令 \end{overlay}
\begin{note} 文本及命令 \end{note}
```

slide 环境是 slides 的主体, 整个环境输出成一页, 并且有页码打印在下面, 如果超长, 则会发出警告. overlay 的内容是用于覆盖在 slide 打印的透明片上的, 因此可有不止一张, 而且应该接在相应的 slide 环境的后面, 它的页码附属于相应的透明片, 因此第 6 页透明片后面的覆盖片的页码是 6-a, 6-b 等等.

相应的透明片及覆盖片应该有相同的内容, 它们的区别就在于哪些部分被打印以及哪些部分不打印, 这是通过命令:

```
\visible          \invisible
```

来实现的. 它们的作用规则与字体命令相同. 在透明片里, 只有少部分内容是不可见 (\invisible) 的, 而覆盖片则一开始就发出 \invisible 命令. 不过这两条命令仅适用于 slides 已定义的字体, 对于新增字体不起作用, 都是可见的, 因此不能使汉字‘隐身’. 不过办法还是有的, 就是把该隐藏的汉字统统改成全角空白 (注意不能用两个半角空白代替), 使得它们虽占位但不显示, 例 13-5-2.tex 就是这样做的. 反之, 例 13-5-1.tex 的第二行中应该隐藏的汉字就给显示出来了.

至于 note 则是用来提醒报告者的, 它的页码具有 6-1, 6-2 的形式.

在 slides 里也可使用 L^AT_EX 的命令:

```
\pagestyle{格式}
```

其中的格式有下面 3 种:

plain 透明片、覆盖片以及注记片的页码都在右下角.

headings 同 plain, 只是当 slides 的 clock 选项被选中时, 注记片的左下角会打印出时间标记, 这也是默认的格式.

empty 不打印页码及对齐线.

\pagestyle 命令应该放在上述 3 个环境之外, 也可用 \thispagestyle 命令规定所在页的格式, 但作用范围只限一页.

当 \documentclass 命令中包含选项 clock 时, 以下两条命令就被激活了:

```
\settime{秒数}      \addtime{秒数}
```

其意义是自明的. 注记片上打印的是这些时间的累计, 而且以分为单位, 以提醒报告者.

当文件中含有许多透明片, 可在导言中利用以下命令选择其中一部分输出:

```
\onlyslides{页码表}    \onlynotes{页码表}
```

前者输出指定页码范围的透明片、覆盖片以及注记片, 而后者仅输出指定页码范围的注记片, 页码表的形式如: 2, 5, 9-12, 15, 必须由小至大顺序排列, 其中可以包含并不存在的页码.

最后我们看一个例子 (13-5-3.tex), 其输出如附图所示.

Sample Viewgraphs

Advantages of slides

- Uses special fonts
- Forces key words instead of long text

Color commands may be used

1

- Supports color layers

as color layers

1-a

Add the overlay only for L^AT_EX 2.09

5 min

1-1

§ 13.6 用 L^AT_EX2HTML 生成 html 文件

如果你想把用 L^AT_EX 写的文章放在网上让大家阅览, 那么 L^AT_EX2HTML 是一个很好的选择. L^AT_EX2HTML 实际上是一个翻译器, 把 L^AT_EX 文件翻译成网页文件, 而且为了适应因特网浏览的特点, 它把一个 L^AT_EX 源文件按章节分拆成一系列网页文件, 并插入许多内部链接, 便于读者前后参看翻阅.

L^AT_EX2HTML 实际上是 perl 的脚本程序, 它有以下特点:

- 在生成 html 文件时, 源文件被按照章节或用户的命令拆分成一组文件, 目录、脚注、索引、参考文献等都是其中的小文件;
- 源文件中凡是 L^AT_EX2HTML 能识别的部分都被转化成 html 的相应结构, 再加上适当的超文本链接, 被输出成 html 文件, 因此汉字文本会按原样输出到生成的 html 文件中, 这是 L^AT_EX2HTML 的一大优点;
- 其他如数学符号、公式等则被送到 L^AT_EX 去转化成 dvi 文件, 再变成 ps 文件, 最终转化成 gif 图像文件, 并被链接到 html 文件的相应位置;
- 表格与插图则根据用户设定的 html 版本 (默认 3.2, 也可设为 2.0 或 4.0) 被转换成 html 的表格或图像;
- L^AT_EX 的标签与引用都被转换成内部的超文本链接;
- 也可以建立外部的超文本链接, 从而使生成的 html 成为因特网上更大的文件系统的一部分;
- 对于不同的输出可以使用不同的文本, 这样可以对 dvi 或 html 文件使用不同的优化策略.

另一方面, L^AT_EX2HTML 也有以下的制约:

- L^AT_EX2HTML 翻译器尚不能适应旧版 L^AT_EX 或经典 T_EX 的部分命令, 因此用户最好使用纯粹的 L^AT_EX 命令, 尽量不要使用 L^AT_EX 以外的自造的、技巧性很强的 T_EX 宏;
- 由于数学符号与公式被转化成图像, 因此在有的浏览器里的显示可能会很难看.

L^AT_EX2HTML 的运行

首先建议你随书光盘里的 Doc\本书例题\l2h.ini 拷贝到你的工作目录 (例如 \mytex. 这个文件包含了默认的配置, 可以保证你能顺利生成 html 文件, 等你有了经验再修改这个配置文件.

如果已经在 WinEdt 的编辑器里建立了一个名为 filename.tex 的源文件, 只要点击工具栏里的 L^AT_EX2HTML 按钮, 就可以执行 L^AT_EX2HTML. 你也可以用执行以下命令来启动 L^AT_EX2HTML:

`latex2html filename`

按默认的设置, `LATEX2HTML` 会在 `filename.tex` 所在的目录里产生一个名为 `filename.html` 的标题网页以及其他相关的文件。

由于生成 `html` 文件时要用到很多超文本链接, 因此在执行 `LATEX2HTML` 前应该先执行一轮 `LATEX` 以及 `makeindex`, 然后再执行第二轮 `LATEX`, 在这两轮执行成功后才执行 `LATEX2HTML`. 这样方能保证源文件的 `TEX` 语法的正确性, 并且生成最新的 `aux` 文件供 `LATEX2HTML` 引用。

当你的一切都感到满意时, 就可以把当前目录里所有后缀名为 `html`, `css`, `gif` 的文件上传到服务器里, 供人浏览。

格式文件 `html.sty` 新增的命令

`LATEX2HTML` 提供了一个格式文件 `html.sty`, 其中定义了几个新的环境, 不过 `\begin{环境名}` 以及 `\end{环境名}` 都应该单独占据一行, 否则容易出错:

`htmlonly` 这个环境里的内容仅当运行 `LATEX2HTML` 输出 `html` 文件时才被处理. 当用 `LATEX` 处理输出 `dvi` 或 `ps` 文件时则被忽略.

`latexonly` 与上面相反, 这个环境里的内容仅当运行 `LATEX` 输出 `dvi` 文件时才被处理. 当用 `LATEX2HTML` 处理输出 `html` 文件时则被忽略.

`rawhtml` 这个环境里的内容当运行 `LATEX2HTML` 输出 `html` 文件时被直接插入输出的 `html` 文件内. 当用 `LATEX` 处理输出 `dvi` 文件时则被忽略.

`html.sty` 也提供了一些新命令:

`\latex{...}` 是 `latexonly` 环境的简化形式.

`\html{...}` 是 `htmlonly` 环境的简化形式.

`\latexhtml{...}{...}` `LATEX` 仅处理第一个括号里的内容, 而 `LATEX2HTML` 只处理第二个括号里的内容.

`\htmlref` 本命令生成一个指向 `LATEX` 标签 `\label{reference}` 的内部超文本链接, 命令的格式是 `\htmlref{文本}{reference}`. 在 `dvi` 输出里只出现“文本”的内容, 而在 `html` 输出里会突出显示“文本”的内容, 并作为一个指向 `LATEX` 标签 `\label{reference}` 的内部超文本链接.

`\htmladdnormallink` 同上, 只是超文本链接指向外部的统一资源定位器 (URL), 例如:

`\htmladdnormallink{文本}{http://www.math.ecnu.edu.cn/index.html}`.

`\htmladdnormallinkfoot` 同上, 只是在输出的 `dvi` 文件里把链接的统一资源定位器 (URL) 打印成脚注.

`\hyperref` 是一个按条件选择输出文本的命令. 命令 `\hyperref{文本1}{文本2}{文本3}{reference}` 的效果是: 在 `html` 输出里会突出显示“文本1”的内容, 并作为一个指向标签 `reference` 的内部超文本链接; 在输出 `dvi` 时以“文本2”作为前置文本, “文本3”作为后置文本, 中间输出标签 `reference` 对应的计数

器值. 例如:

```
\hyperref{格式指南}{格式指南(参见第}{节)}{sec:style}
```

在 dvi 输出文件里表现为

格式指南(参见第 7 节)

而在 html 输出文件里则是“格式指南”并附加一个超文本链接.

`\htmlimage` 使用在被转换成内嵌图像的任何环境里 (例如 `figure` 环境), 用于控制图像的生成. 其参数是用逗号隔开的选项序列:

```
[scale=缩放因子], [external], [thumbnail=缩小因子], [flip=变换选项], [align=对齐方式]
```

选项 `scale` 控制了最终图像的大小; 选项 `external` 使得图像不是内嵌的 (默认情形) 而是通过超文本链接得到的; 选项 `thumbnail` 则用一个略图放在说明中 (`thumbnail` 蕴含了 `external`), 缩小因子确定了略图的大小; `flip` 规定了图像变换的方式, 变换选项可以是: `lr` (左右翻转), `tb` (上下翻转), `xy` (交换 x, y 轴), `r90` 或 `ccw` (逆时针旋转 90°), `r270` 或 `cw` (顺时针旋转 90°), `r180` (中心对称). `align` 确定了对齐方式, 相应的选择有: `top` (顶部对齐), `bottom` (底部对齐), `middle` (竖直方向中间对齐), `left` (左边对齐), `right` (右边对齐), `center` (中间对齐). 默认值是 `bottom`. 例如:

```
\htmlimage{scale=1.5,external,thumbnail=0.2}
```

意为用一个原图 $1/5$ 大小的略图放在文件里, 同时用一个链接指向一个外部图像, 其大小是原图的 1.5 倍.

`\htmladding` 用法为 `\htmladding[位置]{统一资源定位器}`, 它把统一资源定位器 (URL) 所指的图像插入到 html 输出中, 可选项位置可以是 `left`, `right` 或 `center`.

`\htmlcite` 类似于 `\htmlref`, 只是用 `\cite` 取代 `\ref`.

`\htmlrule` 加一条水平直线.

编程指南

建议源文件以下列语句开头:

```
\documentclass{article}
\usepackage{html,makeidx,epsf}
```

这是因为生成 html 时要用到 `html.sty`, `makeidx.sty` 以及 `epsf.sty`.

内部标签

建议给每个章节、公式、插图和表格一个标签, 并且最好使用两级标签, 例如 `\label{fig:text}`, `\label{sec:intro}` 等. 这样做的好处是便于搜索, 也便于文件的维护. 而且网页文件不像纸质印刷品可以前后翻阅, 交叉引用主要通过超文本链接来实现, 因此多加标签及引用更显重要.

外部标签

L^AT_EX2HTML 可以把交叉引用的范围扩大到相关的文件, 这是因为它在与 `html`、`gif` 等输出文件的同一个目录里建立了一个 perl 文件 `labels.pl`, 这个

文件专门存放标签信息. 通过命令:

```
\externallabels{指向外部文件所在目录的统一资源定位器}{此外部文件的
labels.pl 的本地复本}
```

可以建立与外部文件标签信息的联系, 再利用命令:

```
\externalref{外部文件的标签}
```

或

```
\externalcite{外部文件的参考文献标签}
```

作交叉引用. 例如先使用命令:

```
\externallabels{http://www.math.ecnu.edu.cn/doc/manual}{d:/tmp
/labels.pl}
```

指出欲引用的外部网页文件位于目录 `www.math.ecnu.edu.cn/doc/manual` 内, 而那个文件的标签信息在本地机器的 `d:/tmp/labels.pl` 内有一个复本. 现在你就可以利用命令:

```
\externalref{crossrefs}
```

引用那个外部文件里的标签 `crossref`.

为了方便标签管理, 建议把需要外部引用的标签用3级命名, 例如: `doc-label:fig:text`, `doc-label:eqn:text` 等.

章节标题

不要在 `\section` 或 `\subsection` 命令的标题中使用数学模式, 因为这会给 `LATEX2HTML` 的处理带来问题.

注意表达方式

考虑到媒体特点不同, 在网页文件里不宜使用“如前所述”、“后面的”这类定位词, 在 `dvi` 或 `ps` 文件里不应出现“点击这里”的说法.

提供到打印版本的链接

最好能在 `html` 网页第一页的顶部放一个超文本链接, 指向文件完整版本的 `ps` 或 `pdf` 文件, 便于阅览者下载, 而且最好附有文件大小的注解.

添加指向个人主页的链接

在导航板里加上指向作者个人主页的链接是必要的, 这可通过建立 `Home` 按钮实现. 其命令为

```
\htmladdtonavigation{\htmladdnormallink
{\htmladding{按钮图像文件名}}{http://...}}
```

例如:

```
\htmladdtonavigation{\htmladdnormallink{\htmladding
{../home.gif}}{http://www.math.ecnu.edu.cn/~{author/}}}
```

当然你必须为按钮建立一个图像文件(例如这里的 home.gif).

一些注意事项

- 上下标不要和相关的字母分离, 例如不要用 P^2 , 应该用: P^2 .
- 不要对一个 T_EX 命令多次重新定义, 因为 L^AT_EX2HTML 只认识最后一个定义. 此外, 在 T_EX 的命令后面不要用 \ 来产生一个空格, 因为在生成的 html 文件里会把这个 \ 显示出来. 解决方法是使用 ~ 或在 CJK* 里使用 \nbs.
- 请注意 L^AT_EX2HTML 执行时显示的出错信息, 尤其是图像生成或转换时的出错信息. 如果提示你找不到相应的图像文件, 请记住出问题图像的号码, 然后检查 L^AT_EX2HTML 输出目录里的 images.log 与 images.tex 文件, 并设法修改产生错误的 tex 源文件.
- 对 L^AT_EX2HTML 而言, 在数学模式的外部改变字体大小是不起作用的, 例如 a_b 与 $\{\LARGE a_b\}$ 在 html 里的输出是一样的. 为了把字体变大, 可以使用 $\mbox{\LARGE a_b}$.
- 如果你为了避免经常出现水平方向超长的警告, 可以加大 \tolerance 的值, 但是此命令必须放在 latexonly 环境里, 否则它的数值会出现在 html 文件里.

一些模板

L^AT_EX2HTML 的作者指南里提供了一个导言部分的样本模板(参见 latex2html \docs\LaTeX2HTML Authors' Guide.pdf). 我们把它提取出来作为 13-6-1.tex, 并改成中文. 此外, 这个指南也提供了输入插图、公式以及表格的样本模板, 我们都放在例 13-6-2.tex 里, 供读者应用时借鉴. 其中插图的模板有两种, 一种是不同图像格式共用的模板, 这是因为 L^AT_EX 和 L^AT_EX2HTML 要求的图像格式是 eps 或 ps, 而 pdfL^AT_EX 则要求 jpg, pdf, png, tif 格式, 因此为了各种情况都能通用, 对于同一个图像需要提供至少两种格式的文件, 但命令可以用同一个. 另一个模板是对不同格式的图像用不同的命令嵌入, 这两个模板应用时在导言里都要加入: \usepackage{graphicx}.

初始设置文件的修改

L^AT_EX2HTML 在 Windows 系统里的初始设置文件的默认名称是 l2h.ini, 它应存放在当前目录里. 其中的一些初始设置可以被用户修改, 此文件内有详细说明. 我们这里选几个重要的参数加以说明, 等号后面是默认值.

$\$DESTDIR = ''$; L^AT_EX2HTML 输出的 html 等文件的目录, 默认值是与被处理的文件同名的子目录.

$\$NO.SUBDIR = 1$; 置 1 时, 输出文件被放在与输入文件相同的目录里, 置 0 时, 输入文件被放到由 $\$DESTDIR$ 指定的子目录里.

`$NO_NAVIGATION = 0`; 把 0 改成 1 就取消每个网页顶端的导航板.

`$AUTO_NAVIGATION = 1`; 设为 1 时, 自动在每页顶端放一个导航板, 当每页的单词个数超过 `$WORDS_IN_PAGE` (默认值是 300) 时, 则在底部也放一个导航板. 设为 0 则不在底部增设导航板.

`$INDEX_IN_NAVIGATION = 1`; 改为 0 值则取消导航板里指向索引的链接.

`$CONTENTS_IN_NAVIGATION = 1`; 改为 0 值则取消导航板里指向目录的链接.

`$NEXT_PAGE_IN_NAVIGATION = 1`; 改为 0 值则取消导航板里指向下一页的链接.

`$PREVIOUS_PAGE_IN_NAVIGATION = 1`; 改为 0 值则取消导航板里指向上一页的链接.

`$INFO = 0`; 设为 1 则加入一节“关于本文件...”, 设为 0 则取消这一节.

`$SHOW_SECTION_NUMBERS = 0`; 设为 1 则自动显示节号.

`$TITLES_LANGUAGE = "chinese"`; 也可设为 `english`, 这时“目录”、“索引”等标题均用英文显示. 为使导航板的图标也变成中文或英文, 我们在 `latex2html` 目录里放了两个文件, 分别命名为 `l2h.ini.chinese` 与 `l2h.ini.english`, 读者可根据你选用的语言, 把其中之一复制到输入文件的目录里, 并改名为 `l2h.ini`.

分段处理

`LATEX2HTML` 可以把一个大文件分拆成好几个片段 (segment) 分别处理, 每个片断的处理都是独立的, 标签信息都记录在辅助文件中, 修改一个片断只需重新编译这个片断, 除非这个片断的标签发生变化, 而且其他文件要引用这个标签. 此外, 整个文件可以不经修改地由 `LATEX` 编译, 生成印刷版本.

`LATEX` 读入的顶层片断称为父片断, 其他片断称为子片断. 分段工作可以通过以下扩展命令实现:

`\segment{文件名}{章节类型}{标题}`

这个命令标志着一个新的程序段的开始, 此片断位于“文件名.tex”内, 章节类型是指 `LATEX` 的 `\section` 或 `\chapter` 等. 如果你不想让这个片断列入目录, 则可使用其变体 `\segment*`. `LATEX2HTML` 会忽略这两个命令, 而 `LATEX` 则执行以下工作: 首先执行相应的章节命令, 然后 `LATEX` 会把章节计数器与公式计数器的值记录在辅助文件“文件名.ptr”内, 并在这个文件里写入 `\htmlhead` 命令, 告诉 `LATEX2HTML` 如何初始化以处理这个片断文件, 最后, `LATEX` 读入并处理“文件名.tex”.

`\internal[类型]{前缀}`

这条命令使得 `LATEX2HTML` 从文件“前缀-类型.pl”读入指定类型的跨片断信息. 每个片断文件都有唯一的前缀. 而类型必须是以下的一种:

internals 这是默认的类型, 可以不给出. 它输入指定片断文件的内部标签, 使得它们能为当前文件使用.

contents 读入指定片断文件的目录, 使得它们能为当前文件使用.
sections 读入章节信息, 注意包含目录的片断同时需要所有其他片断的目录与章节信息.
figure 读入其他片断的插图表.
table 读入其他片断的表格表.
index 读入其他片断的索引信息.
images 允许其他片断的图像能被当前文件重复使用.

显然, 前缀越短越好, 而默认的前缀是被处理的文件名, 因此子文件的名称越短越好.

```
\startdocument
```

由于 `\begin{document}` 与 `\end{document}` 只能出现在父片断里, 因此不经处理的话, L^AT_EX 不能单独处理子片断. 但是可以被 L^AT_EX2HTML 分开处理, 当然要知道导言部分在哪里结束, 这就是 `\startdocument` 的任务, 它在子片断里起着 `\begin{document}` 的作用, 而被 L^AT_EX 忽略.

```
\htmlhead{章节类型}{标题}
```

此命令不出现在源文件里, 是由 `\segment` 命令自动生成的, 被 L^AT_EX 忽略. 通过辅助文件“文件名.ptr”把父片断的信息传递给子片断, 它告诉子片断当前的章节类型与标题.

```
\htmlnohead
```

本声明出现在文件片断的导言的头部时, 意为从当前页舍弃所有已放置的内容. 通常这是指从 ptr 文件的 `\htmlhead` 传来的章节信息, 但不影响编号及色彩信息. 这使得用户可以改变章节标题或干脆不要标题.

原软件包提供了一个分段处理的例子 `report.tex`, 我们把它汉化修改为 `13-6-3.tex`, 这是父片断, 两个子片断是 `13-6-3s1.tex` 与 `13-6-3s2.tex`, 它们的前缀分别是 `13-6-3s1`, `13-6-3s2`. 分段处理的过程如下: 先用 L^AT_EX 编译 `13-6-3.tex` 两遍; 再用 `dvips` 生成 ps 文件; 最后用 L^AT_EX2HTML 处理: 分别处理 `13-6-3s1.tex` 及 `13-6-3s2.tex`, 由于 `13-6-3s1.tex` 引用了 `13-6-3s2.tex`, 因此还需处理一遍 `13-6-3s1.tex`, 最后处理 `13-6-3.tex`, 大功告成. 由于 L^AT_EX2HTML 处理含汉字的图像有问题, 因此例 `13-6-3s1.tex` 及 `13-6-3s2.tex` 里的英文尚不能改成中文.

例 `13-6-4.tex` 是从 T_EX4ht 的例 `13-7-1.tex` 改编的, 读者可与 T_EX4ht 生成的网页比较, 不过有一点是很清楚的: L^AT_EX2HTML 与 `colortbl` 包不兼容, 因此生成的表格没有彩色, 而且把 `yellow` 等打印出来了.

关于 L^AT_EX2HTML 的更多功能, 请见 `latex2html\docs\LaTeX2HTML Authors' Guide.pdf` 以及 `latex2html\docs>manual.ps`.

§ 13.7 用 T_EX4ht 生成 html 文件

我们先看一个例子 (13-7-1.tex):

```
\documentclass[12pt]{article}
\usepackage{colortbl}
\usepackage{CJK}
\begin{document}
\begin{CJK}{GBK}{song}
\author{某作者}
\title{一些数学问题}
\maketitle
\renewcommand{\contentsname}{目录}
\tableofcontents
\section{第一节}
\subsection{第一小节}
一个方程:  $x^2+y^2=z^2$ .

\noindent 另一个方程:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y}$$

\subsection{第二小节}
一个表格:
\begin{tabular}
{|>{\columncolor{yellow}\bfseries}l|>
{\columncolor{red}}p{4cm}|}
A&1\\\hline
B&2\\
\end{tabular}
\section{第二节}
\section*{第三节}
\end{CJK}
\end{document}
```

只要按一下工具栏里的 T_EX4ht 键, 就在源文件所在目录里生成 13-7-1.html 等文件. 只需把生成的后缀名为 html, css, gif 的文件上传到服务器, 就能供人浏览了.

也可以使用命令行的方式, 在命令提示符后面键入命令:

```
htlatex 13-7-1 "参数一" "参数二" "参数三"
```

就能生成文件 13-7-1.html. htlatex.bat 除了调用 L^AT_EX 外, 还调用了 tex4ht 以及 t4ht 两个程序. tex4ht 生成 html 文件, 而 t4ht 则生成数学公式所需的图形文件.

参数一是供 L^AT_EX 用的, 下面有介绍; 参数二是供 tex4ht 用的, 指示从 ht-fonts 的根目录到子目录的路径, 在一开始设定好后就不必再设置了; 参数三是供 t4ht 用的, 唯一可用的值是 -p, 它告诉 t4ht 不要生成图形, 当你已经生成了所需的图形文件, 以后只是修改文本或格式时, 选用此参数可以节省时间. 如果前两个参数为空, 则应使用 "" "" "-p" 的形式.

参数一的常用取值是

html 生成 html 文件, 这是默认值.

2 生成的 html 按目录的二级标题 (\section) 分割成多个文件, 这个选项的取值范围从 1 (按 \chapter 分割) 直至 4 (按 \subsubsection 分割). 没有此选项时, 总是生成单独一个 html 文件.

section+ 建立从目录到相应章节的链接.

frames 使目录与正文分别放在两个框架内.

参数一的第一项一般是扩展名为 cfg 的配置文件名, 其格式见后面说明, 因此不使用配置文件时, 第一项应为 html, 例如: "html,2,sections+". 读者可选用各种不同参数编译 13-7-1.tex, 以观察其效果.

另一种方法是建立一个配置文件, 它的扩展名必须是 cfg, 格式为

```
前导定义
\Preamble{选项}
... 导言定义 ...
\begin{document}
... 头部定义 ...
\EndPreamble
```

其中的选项就是上面提到的参数一, 例如 \Preamble{html,2,section+} 相当于 \usepackage[html,2,section+]{tex4ht}. 前导定义与导言定义都插在源文件的导言中, 分别位于读入宏包 tex4ht 命令的前后, 而头部定义则放在生成的 html 文件的头部. 这些定义中可用到下列的 T_EX4ht 的底层命令. 使用配置文件的好处是可以构造出各种复杂格式的 html 文件, 而不需更动 L^AT_EX 的源文件. 我们这里附了两个配置文件: htlatex1.cfg 与 htlatex2.cfg, 前者很简单, 后者则建立了框架结构, 试分别用这两个配置文件编译 13-7-1.tex, 并作比较. 读者会发现使用框架结构时左边有两个 Contents, 只要把 13-7-1.tex 中的 \tableofcontents 注解掉就可解决.

T_EX4ht 的几个底层命令

\HCode{...} 直接插入 html 命令, 例如 \HCode{<HR>} (画一条水平直线).

\HPage{锚名}内容\EndHPage{} 建立一个超文本页, 它可以通过锚名进入, 可惜这里的锚名似乎不接受汉字. 含在内容中的命令 \ExitHPage{锚名} 会创立一个退出按钮. 例如:

```
\HPage{Enter another page}
..... [\ExitHPage{Exit this page}] .....
\EndHPage{}
```

`\Link[目标文件]{目标位置}{当前位置}锚名\EndLink` 建立一个链接到‘目标文件#目标位置’的锚, 而且为当前位置建立标记. 当选项目标文件为空时, 默认值是当前文件所创建的文件. 特殊字符~, _, %应分别输入为: ‘\string ~’, ‘\string _’以及‘\%’. 例如:

```
\HPage{}
.....
\Link[http://www.tug.org/]{XX}\TeX{} Users Group Home
Page
\EndLink
.....
\EndHPage{}
.....
\Link{XX}{...}\EndLink
```

`\ifHtml... \else... \fi` 根据html状态与非html状态分别加入不同内容, 当非html状态的内容为空时, 可省去`\else`. 例如:

```
\ifHtml \HCode{<HR>} \else \hrule \fi
```

与图形有关的命令

`\Picture[说明文字]{图形文件名}` 指向一个图形文件. 例如:

```
\Picture[**OSU logo**]{http://www.cis.ohio-state.edu/images/
OSU.gif}
```

`\Picture+[说明文字]{文件名_属性}内容\EndPicture` 根据所给的内容作图, 并存储在一个图形文件中, 同时指向此图形文件. 当文件名为空时, 使用自动生成的文件名, 不过当后面有属性时, 第一个字符应是空格. 例如:

```
\Picture+[ align="right"]Text within a picture.\EndPicture
创建一个以文本为内容的图形.
```

`\Picture*[说明文字]{文件名_属性}内容\EndPicture` 与带+号的命令类似, 只是生成一个竖直的盒子.

我们把上面的一些命令加入后做了一个例13-7-2.tex, 请读者用不同配置编译比较.

关于T_EX4ht的详细说明请参看texmf\doc\html\tex4ht\mn.html.

附录一 TeX 系统的安装

随书光盘的内容十分丰富,除了有 TeX 系统的安装程序以及本书例题的源文件外,还有一些应用软件、使用经验和模板等,务请大家先看一下光盘中的文件 ChinaTeX-CD.txt,里面有光盘的内容介绍。

对于心急的读者,建议你在安装前先阅读 A1.1; 对于高级的读者,可以考虑手工安装,则请直接阅读 A1.3. 许多更进一步的内容可参见随书光盘里的文件几点使用说明.txt, 手工安装指南.txt, 版本更新记录.txt.

A1.1 快速安装

为了成功安装,首先请把以前安装在机器上的 MiKTeX 或 ChinaTeX 系统卸掉,同时把你个人的文件备份好. 安装程序可以自动生成 6 款中文字体: 宋体、黑体、仿宋体、楷体、隶书以及幼圆. 但是生成这些字体要利用 Windows 的 fonts 目录里的 ttf 字库,一般简体中文 Windows 里有宋体和黑体 2 款 GBK 字库 simsun.ttc¹ 及 simhei.ttf. 如果按缺省配置安装了简体中文 Office, 则会自动安装另外 4 款字体: stkaiti.ttf、stfangso.ttf、simli.ttf、simyou.ttf, 这样你的 6 款字体就齐了. 如果你没有安装简体中文 Office, 那么除非你自己设法找到这 4 字库并把它们复制到 Windows 的 fonts 目录里, 否则就只能安装宋、黑两款字体.

做好上述准备后,你就可以动手安装了. 把随书光盘放入驱动器,一般会弹出安装窗口. 如果没有反应,就要手工执行光盘上的 Setup-ChinaTeX.exe. 在按了两次回车后,出现了“选择组件”的窗口,上面的红勾都是可选的组件,6 种字体中没有打勾或变灰的字体就是缺少 ttf 字库文件的字体. 建议你吧能安装的全部选中,再进入下一步.

在接下来的“TeXMF 根目录”里,有一个建议目录,你除了修改盘号外,最好不要改动. 进入下一步,就弹出一个 DOS 窗口将文件解压缩到选定的 TeXMF 根目录. 然后又要你选择“目标文件夹 (Local TeXMF 根目录)”,同样希望你尽量不改动. 点击“下一步”,让其完成全部安装.

安装完成后,请你再建立一个安放 TeX 源文件的工作目录,比方说 mytex. 然后把文件 texmf\doc\latex\CJK\chinese\test-GBK.tex 拷贝到 mytex. 点击桌面上的 WinEdt 图标,就可以测试了. 在 WinEdt 里打开上述文件,里面列举了 6 款中

¹在 Windows 9X 平台上为 simsun.ttf.

文字体, 把你没有安装的字体系删去. 然后点击工具条 (就是有许多图标横条) 上的 L^AT_EX 图标, 进行编译, 完成后再点击蓝色放大图标 (下注 DVI 的那个), 等它生成好字体后, 就会显示排版结果, 如果一切正常, 就说明你的 L^AT_EX + CJK 系统安装成功了.

如果在上述过程中出现找不到文件的警告或出错信息, 就请你点击工具条中间偏左的两个齿轮的图标, 这时会弹出名为 “MiKTeX Options” 的窗口. 最上面的长方块名为 “File name database”, 里面有一个名为 “Refresh Now” 的按钮. 点击此按钮, 让 MiKTeX 更新它的文件名数据库, 完成后按 “确定” 关闭窗口. 再重新编译、预览, 也许就成功了. 以后凡是遇到找不到文件的警告, 都可以这样做一遍, 把数据库更新.

如果你的安装不成功, 建议你干脆把它卸载 (点击 “开始 → 所有程序 → ChinaTeX → 卸载 ChinaTeX”), 清除掉以前安装的 T_EX 系统留下的垃圾文件, 再重新安装, 说不定就成功了. 如果你的机器太陈旧, 或者使用过时的操作系统, 都有可能安装不成功, 而且可能是无法自动安装的. 这时只好采用下面所说的手工安装, 并且放弃某些功能. 另一个失败的原因往往来自卸载不干净的 T_EX 系统的作祟, 或者是你要自作主张不肯接受默认的目录所导致. 只要你耐心除去注册表里的垃圾, 还是可以安装成功的.

此外, 如果你要生成 pdf、html 文件的话, 除了 T_EX 系统本身外, 往往需要外围软件的配合, 例如 GhostScript、GSview、Perl、ImageMagick 等, 而且往往需要最新的版本. 因此在你遇到问题时, 请先检查一下你所安装的上述软件版本, 与随书光盘 Extras 下收录的版本比较一下, 或者干脆卸去旧版, 改装新版, 说不定就成功了.

以下内容专供高级用户参考.

请注意, 如果你没有卸载原有的 T_EX 系统, 那么 Local TeXMF 的目录就是你原来使用的目录, 不能更换. 此外, 安装程序会自动记录 Local TeXMF 目录下的几个可能被用户修改的配置文件的 MD5 哈希值:

```
dvipdfm\config\cid-x.map,  
dvips\config\config.ps,  
miktex\config\miktex.ini,  
miktex\config\updmap.cfg,  
ttf2tfm\base\ttf fonts.map,  
ttf2tfm\base\ttf shape.map.
```

如果你改动了它们的内容, 它们将不会被卸载程序删除. 卸载程序也不会删除绝大多数安装后自行增加的文件, 但是为了保险起见, 我们还是建议你在卸载之前对重要的文件进行备份.

如果你的操作系统是 Win9X/Me, 那么使用随书的安装盘 (即 ChinaTeX) 可能遇到的问题如下:

1. ImageMagick 的最新版本 6.2.x 在 Win9X/Me 操作系统下运行存在问题, 可能影响某些系统功能. 它的一个低版本 5.5.7 虽然没有这个问题, 和 Ghostscript 的配合却有问题, 结果造成 T_EX4ht 无法进行图像格式转换. 目前 ChinaTeX 为

Win9X/ME/NT/2000/XP 配置的全部是安装 6.2.x, 如果你必须使用 Win9X/ME 平台, 请自行承担可能带来的任何风险, 你也可以自行尝试使用 ImageMagick 和 Ghostscript 的不同版本, 请设法找到一个可以匹配的组合 (由于 Ghostscript 的版本太多, 我们没有做这个工作, 如果你找到解决方案, 请联系我们将其加入 ChinaTeX 的后续版本).


2. 根据我们的测试, $\text{\LaTeX}2\text{HTML}$ 在 Win9X/ME 下似乎无法正常工作, 可能造成 DOS 窗口死机, 我们建议你使用 $\text{\TeX}4\text{ht}$ 进行 \TeX 到 HTML 的转换. 如果你找到了在 Win9X/ME 下使用 $\text{\LaTeX}2\text{HTML}$ 的办法, 请与我们联系, 以将你的方案加入 ChinaTeX 的后续版本.

3. 当你将 ActivePerl、ImageMagick 安装到有空格的目录中时, 其安装程序不能正确加入系统搜索路径, 你可以手工修改 Autoexec.bat 文件, 在 PATH= 后面的路径两侧加上引号, 并删除 ChinaTeX 安装程序可能插入的重复路径.

4. 由于 Win9X/ME 的初始环境变量空间可能不够大, 从而造成安装后 MiKTeX 的某些功能无法正常运行, ChinaTeX 的安装程序会自动增大环境变量空间到 4096 字节, 你可以修改 ChinaTeX.ini 中的 EnvSize_9X 指定缺省值. 增大环境空间变量需要修改你机器上的 Config.sys 文件, 该文件将在卸载 ChinaTeX 后会自动恢复. 如果你在使用 ChinaTeX 期间更改了这个文件, 则卸载 ChinaTeX 后你需要手工从 Config.sys.bak 文件恢复你的改动.




A1.2 安装测试

我们的随书光盘集成了多个系统, 读者可用盘中的一些文件对自己感兴趣的系统作测试. 以下的 “[X:\]” 代表随书光盘所在的光盘驱动器根目录, “texmf” 则是你在硬盘建立的 TeXMF 根目录.

测试时, 先把文件拷贝到一个工作目录 (例如 \mytex) 里, 然后点击桌面上的 WinEdt 图标  , 打开 WinEdt, 读入测试文件, 就能进行测试.

1. 测试 CJK, 请尝试编译

texmf\doc\latex\CJK\chinese\test-GBK.tex

打开文件后, 请删去那些你没有安装的字体的, 再用鼠标左键点击工具条 (即有各种图标的横条) 上的保存文件按钮  (如果没有修改就不必按此键), 然后点击 \LaTeX 按钮  , 对窗口里的文件进行编译, 完成后, 点击预览按钮  , 等程序生成点阵字体后, 就会弹出窗口显示排版的结果. 如果没有显示, 请看看屏幕下方的有没有名为 Yap 方框, 若有, 点击它就会弹出显示窗口.



类似地尝试编译

texmf\doc\latex\CJK\chinese\test-UTF8.tex

可以测试 UTF8 编码功能.




2. 测试 pdf \LaTeX , 请尝试编译随书光盘里的

[X:\]Doc\本书例题\13-4-1.tex

打开文件后, 点击 pdf \LaTeX 按钮 , 对窗口里的文件进行编译, 由于本文中间要引用后面列出的参考文献, 因此需要编译两次. 完成后, 点击 Acrobat Reader 按钮 , 就会显示漂亮的演示文件, 按 Ctrl-L 可以切换窗口或全屏显示. 这就是 beamer 宏包的例子.



对加强 pdf 演示效果的 PPower4 有兴趣的读者可以尝试编译随书光盘里的

[X:\]Doc\常用中文文档\PPower4中文手册(汤银才 20050508)\old\PPower4的使用.tex

打开文件后, 点击 pdf \LaTeX 按钮 , 对窗口里的文件进行两次编译, 再点击 PPower4 按钮 , 对生成的 pdf 文件作后处理, 完成后, 点击 Acrobat Reader 按钮 , 就会显示漂亮的演示文件.



3. 测试 $\text{\LaTeX}2\text{HTML}$, 请尝试编译随书光盘里的

[X:\]Doc\本书例题\13-6-4.tex

请注意把同一目录中的 12h.ini 一起拷贝过去. 打开文件后, 点击 $\text{\LaTeX}2\text{HTML}$ 按钮 , 对窗口里的文件进行编译(注意至少重复做两遍), 完成后, 点击 $\text{\LaTeX}2\text{HTML}$ 按钮下面的网页浏览按钮 , 就会显示生成的 html 文件. 再请点击网页中的“第一小节”, 看到两个数学公式被正确显示了, 才算大功告成. 由于 $\text{\LaTeX}2\text{HTML}$ 版本较旧, 很久没有更新, 因此运行不太稳定. 如果在你的机器上安装不成功或经常出问题, 建议你改用 $\text{\TeX}4\text{ht}$.


4. 测试 $\text{\TeX}4\text{ht}$, 请尝试编译随书光盘里的


[X:\]Doc\本书例题\13-7-1.tex

打开文件后, 点击 $\text{\TeX}4\text{ht}$ 按钮 , 对窗口里的文件进行编译, 完成后, 点击 $\text{\TeX}4\text{ht}$ 按钮下面的网页浏览按钮 , 就会显示生成的 html 文件. 注意观察其中两个数学公式是否被正确显示.

5. 测试天元软件, 请尝试编译随书光盘里的

[X:\]Doc\本书例题\test.ty

打开文件后, 点击天元 TY 按钮 , 对窗口里的文件进行编译并显示排版结果.

对于有些习惯使用天元 + \LaTeX 的老用户, 可以按以下步骤把天元按钮改为“TY+AMSTeX”: 选中“Options → Menu Setup”, 点击鼠标左键后会弹出一个名为“Menu Setup”的窗口, 在左下角的小窗口里选中“&Accessories”, 连击鼠标左键两下后又弹出一个名为“Menu Setup: &Accessories”的窗口, 在左下角的小窗口里选中“T&Y AMSTeX”, 让蓝条停留在其上, 在中间的下方有一栏名为“Images and Hint”, 下左方是一个空白大方块, 在方块里面点击鼠标左键后, 又弹出一个名为“Tool Bar Icons”的窗口, 在其中找到“天元 TY”图标 , (应该在最下方), 在其上点击鼠标左键后, 看到图标出现在上方了, 就按“OK”关闭窗口, 再连按两次“OK”, 就大功告成. 现在这个天元按钮就相当于“天元 + AMSTeX”了. 用同样方

法可使其恢复原来的功能。



现在可把随书光盘中的测试文件

[X:\]Doc\本书例题\test_ams.ty

拷贝到 \mytex 目录里, 用同样方法测试“天元 + AMSTeX”。



6. 测试旧版 CCT, 请尝试编译

texmf\cct\doc\old-CCT-sample.ctx

打开文件后, 点击“旧 CCT”按钮 , 对窗口里的文件进行编译, 完成后, 点击预览按钮 , 显示排版的结果。



7. 测试新版 CCT, 请尝试编译

texmf\cct\doc\README.tex

不过要把同一目录里的图像文件 cctdiag.eps 一起拷贝过去。打开文件后, 点击“汉 TeX”按钮 , 对窗口里的文件进行编译, 完成后, 点击预览按钮 , 显示排版的结果。




8. 测试 ConTeXt, 请尝试编译

texmf\doc\context\chinese\test.tex

打开文件后, 请删去那些你没有安装的字体, 存盘后再点击“ConTeXt”按钮 , 对窗口里的文件进行编译, 这时会弹出一个要求输入参数的窗口, 如果你想直接输出 pdf 文件, 可以输入参数 --pdf, 如果是先生成 dvi 然后再用 dvipdfm 转化, 则不需输入任何参数。请注意: 输入的参数会自动记忆。完成后, 点击预览按钮 , 显示排版的结果。





9. 测试 Omega-CJK, 请尝试编译

texmf\doc\omega\chinese\test.tex

打开文件后, 请删去那些你没有安装的字体, 存盘后再点击“Omega-L^AT_EX”按钮 , 对窗口里的文件进行编译, 完成后, 点击预览按钮 , 显示排版的结果。不过转换成 ps 文件时, 要点击“DVIPS for Omega”按钮 , 而不是通常的“DVIPS”按钮。

10. 测试 ttshape 和孙文昌开发的其他宏包, 请尝试编译

texmf\doc\latex\ttf-MiKTeX\ttfshape

下的几个例子文件。打开文件后, 点击“ttf-L^AT_EX”按钮 , 或者“ttf-pdfL^AT_EX”按钮 , 对窗口里的文件进行编译, 完成后, 点击预览按钮 , 或者 Acrobat Reader 按钮 , 显示排版的结果。

此外为了测试孙文昌开发的 picture.sty 的效果, 请尝试编译

texmf\doc\latex\ttf-MiKTeX\picture\picture.tex

不过需要把这个目录下的所有文件都复制到 mytex 目录里, 然后再用 ttf-L^AT_EX 或 ttf-pdfL^AT_EX 编译。也许需要编译两次, 就能得到美丽的图案。

下面介绍 WinEdt 里两个有用的按钮:



孙文昌开发的 L^AT_EX 辅助输入工具 T_EXfriend, 非常方便.



进入 MiKTeX Options, 点击 “Refresh Now” 即可更新 File Name database. 凡是在 MiKTeX 里添加了新内容或者出现找不到文件时, 都可更新此数据库, 也许就解决问题了.

A1.3 手工安装

尝试手工安装, 将有利于你熟悉和掌握 MiKTeX 及外围相关软件的工作方式, 从而在本光盘套装的基础上自行定制自己喜好的安装模式. 在下面的安装说明中, “[X:\]” 代表随书光盘所在的光盘驱动器根目录, texmf 则是你在硬盘建立的 TeXMF 根目录.

如果你没有 7z 的解压缩软件, 则请进入 [X:\]Extras\7-Zip 子目录, 先安装好此软件.

1. 如果 texmf 目录不存在, 则请将 [X:\]Data\texmf.7z 文件解压到你设定的 TeXMF 目录, 例如 C:\ChinaTeX\texmf, 然后将该目录下的子目录 texmf\miktex\bin 加入系统路径. 如果已经有 texmf 目录, 你也不想把原来的 MiKTeX 系统更新的话, 则直接将 texmf\miktex\bin 加入系统路径即可. 如果你想用随书光盘的最新版替换你原来的 MiKTeX, 则请把原来的系统删除, 再把 [X:\]Data\texmf.7z 文件解压到你设定的 TeXMF 目录.

2. 创建你要安装本地文件的路径, 这里假设为 C:\MiKTeXDirect. 设置系统环境变量 MD_LOCALROOT 为该路径.

3. 将 [X:\]Data\MiKTeXDirect.7z 解压缩到 C:\MiKTeXDirect 目录, 然后将 C:\MiKTeXDirect\miktex\bin 加入系统搜索路径.

4. 使用 xGBKFonts 加入中文字体, 使用方法参考 texmf\miktex\bin\xGBKFonts.bat.

例如使用以下命令:

```
xGBKfonts.bat simsun.ttc song
xGBKfonts.bat simhei.ttf hei
xGBKfonts.bat stkaiti.ttf kai
xGBKfonts.bat stfangso.ttf fs
xGBKfonts.bat simslit. ttf li
xGBKfonts.bat simyou.ttf you
```

5. 在命令行执行 initexmf -u、initexmf --mkmaps 及 initexmf -u, 即可开始使用 MiKTeX.

6(可选). 如果你要使用 netpbm 的完全版 (L^AT_EX2HTML 需要 netpbm 支持), 请将 texmf\netpbm\bin 加入系统搜索路径. MiKTeX 中附带了少量同名的 netpbm

程序, 我们已将其改名为 xxxxxx-miktex.exe 的格式, 以避免重名. 如果你想使用 MiKTeX 的相关命令, 请自行修改.

7(可选). 如果你要使用 PPower4, 需要先安装 Java2RE ([X:\]Extras\Java2RE-SE 目录下有最新版本的安装程序), 然后请参考

C:\MiKTeXDirect\miktex\bin\ppower4.bat.org

在 C:\MiKTeXDirect\miktex\bin 目录下建立一个文件 ppower4.bat.

8(可选). 如果你要使用 L^AT_EX2HTML, 请将

[X:\]Data\LaTeX2HTML-2002-2-1.7z

解压缩到 C:\LaTeX2HTML 目录, 并将 C:\LaTeX2HTML\bin\latex2html.bat 转移到任一个系统可以搜索到的路径(如果你的 L^AT_EX2HTML 安装路径不是 C:\LaTeX2HTML, 则需要增加一个系统环境变量 LATEX2HTMLDIR 指向该路径).

9(可选). 如果你要使用 WinEdt, 请将

[X:\]Data\WinEdt-5.4SP1-20050701.7z

解压缩到任一个目标路径, 然后在该路径中执行

WinEdt.exe "[Exe('%B\WinShell\Install.edt'))];]"

如果 WinEdt 的安装目录不是 C:\ChinaTeX\WinEdt, 请修改安装目录下的 edt.bat 文件中的路径.

10. 其他外围工具的安装: ActivePerl、Java2RE、AFPL Ghostscript、ImageMagick、Adobe Reader、GSview 等常用程序的最新版安装程序可以在随书光盘的 [X:\]Extras 目录下找到, 注意前四个程序是不少 T_EX 组件所必需的底层支持环境, 如果没有安装将无法使用相应的功能.

还有几点说明:

1. texmf 目录占用大约 1GB 的硬盘空间, 其中的子目录 doc 和 source 大约占到 40%, 如果你的硬盘空间有限, 可以将 source 目录和 doc 目录下不常用的文档删除.

2. 用 xGBKFonts.exe 生成全部字体所需的时间较长, 为了方便你以后的重复安装, 你可以将第一次安装生成的字体文件和相关配置文件另行保存, 以后安装时可以不选择中文部分, 在安装后将相关文件直接拷贝到安装目录即可. 备份中文字体文件请执行 backup-GBKFonts.bat 文件, 你可以自行修改该文件备份更多内容.

3. texmf\fonts\truetype 目录中提供了 bitstream 公司的免费 Unicode 大字符集字体 cyberbit.ttf (位于 bitstrea 目录下). 如果你希望在 Windows 的其它程序中使用这款字体, 请在资源管理器界面下将它复制到“控制面板 | 字体”目录下 (在命令行下使用 copy 命令不能在 Windows 中注册字体).

4. texmf\fonts\truetype\arphic 目录下提供了两款文鼎公司的免费 GB 中文 TTF 字体, 以供你测试 ttshape 之用. 你可以资源管理器界面下将它复制到“控制面板 | 字体”目录下, 以在其它程序中使用这两款字体.

5. 在你将 cyberbit.ttf 和文鼎公司的 GB 中文字体复制到“控制面板 | 字体”的情况下, 可以将这 3 个文件从 texmf\fonts\truetype 删除, 在定制 ChinaTeX 时可用于节省空间.

A1.4 如何制作自己的 MiKTeX Direct

MiKTeX Direct 的本来含义是把全部 T_EX 系统文件放在任何目录下, 只需将 `texmf\miktex\bin` 加入系统搜索路径, 即可马上开始使用 T_EX. 当 `texmf` 目录位于光盘的时候, 就称为“MiKTeX Direct CD”, 实际上这也是目前 MiKTeX 官方唯一的发行方法, 但是你完全可以将 `texmf` 放在硬盘上运行, 这时实际上我们得到的就是一个“MiKTeX Direct HD”系统. MiKTeX Direct 的设计使得安装得到极大的简化, 用户无需操作相对复杂的注册表, 而只需增加系统搜索路径, 这使得 MiKTeX 系统可以自由地在不同机器间进行移植而无需执行专门的安装程序.

ChinaTeX 实际上是 MiKTeX Direct 的一次再包装, 把外围的程序和需要补充的中文部分用同一个安装程序进行打包, 以方便用户使用尽可能完整的 T_EX 功能. 但是, 由于字体版权的原因, ChinaTeX 的发行介质上不能包含需要的中文 GBK 字体 (目前已有的免费字体还不能满足大部分用户的实际需求), 因此对于 ChinaTeX 而言, 中文部分必须通过安装程序现场生成, 或者使用用户自行备份的文件. 另外, 对于 ChinaTeX 的 CD 版, 还需要将 `[X:]Data\texmf.7z` 解压到指定的 TeXMF 目录. 这些都从某种程度上抵消了 MiKTeX Direct 设计上带来的方便移植的好处.

为了最大程度地发挥 MiKTeX Direct 的优点, 用户可以自己对 `texmf` 进行定制, 将需要的增补文件全部加入到 `texmf` 目录中, 将自己认为不需要的文件从 `texmf` 删除. 下面分类介绍一下如何向 `texmf` 中增加文件.

对于不涉及字体的普通 T_EX 文件, 你只需执行如下两个步骤:

1. 将文件拷贝到 `texmf` 下的相关目录.
2. 再执行一次 `initexmf -u` 命令, 然后将

`C:\MiKTeXDirect\miktex\config\texmf1.fndb`

复制到 `[X:]texmf\miktex\config`, 并改名为 `texmf.fndb` (需要删除已存在的文件).

对于一般的 T_EX 字体文件, 你需要执行如下几个步骤:

1. 将文件拷贝到 `texmf` 下的相关目录.
2. 如果增加的内容包含 `map` 文件, 并且你希望直接使用其中的字体 (而不是通过具体命令调入), 则在 `[X:]texmf\web2c\updmap.cfg` 中增加相应的行, 否则直接转到最后一步.

3. 执行一次 `initexmf -u`.

4. 执行 `initexmf --mkmaps` 命令, 然后将

`C:\MiKTeXDirect\dvipdfm\config`

`C:\MiKTeXDirect\dvips\config`

`C:\MiKTeXDirect\pdftex\config`

目录下的 `*.map` 文件复制到 `[X:]texmf` 中的相应目录 (覆盖原有文件).

5. 再执行一次 `initexmf -u` 命令, 然后将

`C:\MiKTeXDirect\miktex\config\texmf1.fndb`

复制到 `[X:]texmf\miktex\config`, 并改名为 `texmf.fndb` (需要删除已存在的文件).

对于涉及 ttf 字体的 \TeX 文件, 则一般还需要修改部分文件提供 ttf 文件的信息. 这里以 xGBKFonts 生成的 GBK 中文字体为例, 介绍相关的步骤.

1. 将制作好的中文文件中的 `ttf2tfm\base\ttfonts.map` 文件中的内容追加到 `[X:\]texmf\ttf2tfm\base\ttfonts.map`.

2. 将除 `ttf2tfm` 目录以外的其他目录整个拷贝到 `[X:\]texmf`.

3. 如果你的 `[X:\]texmf\web2c\updmap.cfg` 中没有下面这行内容:

```
Map cjk.map
```

则增加之.

4. 执行一次 `initexmf -u`.

5. 执行 `initexmf --mkmaps` 命令, 然后将

```
C:\MiKTeXDirect\dvipdfm\config
```

```
C:\MiKTeXDirect\dvips\config
```

```
C:\MiKTeXDirect\pdftex\config
```

目录下的 *.map 文件复制到 `[X:\]texmf` 中的相应目录 (覆盖原有文件).

6. 再执行一次 `initexmf -u` 命令, 然后将

```
C:\MiKTeXDirect\miktex\config\texmf1.fndb
```

复制到 `[X:\]texmf\miktex\config`, 并改名为 `texmf.fndb` (需要删除已存在的文件).

现在如果你把 `[X:\]texmf` 目录里的内容刻录在光盘上, 你自己的 MiKTeX Direct CD 就制作成功了.

A1.5 转换为 MiKTeX Direct 工作模式

如果你已经安装了普通的 MiKTeX 系统, 如 CTeX 中文套装, 可以通过如下办法将其转换为 MiKTeX Direct 模式.

1. 在 `texmf\miktex\config` 目录下创建一个名为 `md.ini` 的只读文件, 内容请参考任何一个 MiKTeX Direct CD 或者 ChinaTeX 的发行包里面的该文件 (该文件位于随书光盘的 `Data\texmf.7z` 压缩包中, 安装后位于 `texmf\miktex\config` 目录下).

- 2(可选). 如果你希望继续使用 `localtexmf` 下的内容, 请设置环境变量 `MD_LOCALROOT` 指向该目录. 如果不想设置环境变量, 也可以将 `localtexmf` 下的文件拷贝到 `%SystemDrive%\MiKTeXDirect` 目录 (对 WinNT/2k/XP 系统), 或者 `%windir%\MiKTeXDirect` 目录 (对 Win9x/ME 系统).

将普通的 MiKTeX 系统转换为 MiKTeX Direct 模式的好处是: 你可以将 `texmf`、`localtexmf` 两个目录保存在你的移动硬盘上, 免去将来重新安装的麻烦.

附录二 L^AT_EX 命令简介

本附录包含了所有 L^AT_EX 命令的简介, 命令均按 ASCII 字母次序排列, 但出现在第一位的 \ 不参加排序.

有的命令后面带有记号, 其意义如下:

[a] 是属于 *AMS-L^AT_EX* 的命令, 为使用此类命令, 须在前言部分输入宏包 `amsmath`;

[m] 是只允许出现在数学模式中的命令;

[p] 是只允许出现在导言 (preamble) 中的命令.

_ 产生一个正常的空格, 可用在不含参数的命令后面或不表示句子结束的句点后, 以产生一个正常的空格.

! 用在 \index 命令中作为条目分隔符, 例如 \index{command!fragile} 可产生一个索引, 其主条目是 'command', 副条目是 'fragile'.

! ' 产生 j.

\! [m] 在数学模式中产生一个长度为 $-1/6$ quad 的空格 (即后退这段距离): 例如 `xx\!x = xxx`.

" 1. 在正常的文本模式中产生一个右双引号";

2. 用于 MakeIndex 中时, 表示按字面意义打印紧跟在后面的特殊字符 !, @, | 或 ", 例如: \index{"!} 表示产生字符 ! 而不把它看成条目分隔符;

3. BibT_EX 中文本域的分界符, 例如: AUTHOR = "Donald E. Knuth".

\" 产生重音号 umlaut: \"{a} = ä.

在用户自定义命令或环境时被用作变量替换符.

在嵌套的用户自定义命令或环境时被用作内部变量的替换符.

\# 产生字符 #.

\$ 用于在文本模式与行内数学模式之间进行切换的开关字符, 它的首次出现 (从文本到数学) 相当于 \ (或 \begin{math}, 它的再度出现 (从数学到文本) 相当于 \) 或 \end{math}.

- \\$ 产生美元号 \$.
- % 注解符号, 从这个字符开始直至行末的字符以及下一行行首的空白字符均被 T_EX 忽略.
- \% 产生百分号 %.
- & 在 array 以及 tabular 环境中用来表示开始一个新的列.
- \& 产生字符 &.
- \' 1. 产生一个重音号: \'a = á;
2. 在 tabbing 环境中, 表示跳到当前列的末尾, 使此命令左边的文本右对齐.
- () 在 picture 环境的绘图命令中用于指定用一对数表示的坐标; 在 BibT_EX 中可用于条目类型的最外层分组的符号 (通常用的符号是 { }).
- \(从文本模式转换成行内数学模式的开关符, 其作用相当于 \begin{math} 或文本模式下的 \$ 号.
- \) 从行内数学模式转换回文本模式的开关符, 其作用相当于 \end{math} 或数学模式下的 \$ 号.
- \+ 在 tabbing 环境中表示从下一行开始左边的边界移到下一制表位 (跳过一列), 这个命令只能出现在每行的头或尾.
- \, 产生一个长度等于 1/6 quad 的小间隔, 可用于文本或数学模式, 例如: $xx\,,x=xx\,x$.
- 产生连字符 -; -- 产生数字间的连接号 (en dash) -; --- 产生西文破折号 (em dash) —.
- \- 1. 表示隐藏的连字符, 如果在一个西文单词中间含有 \-, 那么这个词的正常断字换行规则将失去作用, 当需要断字换行时, 只能选在出现命令 \- 的位置, 并插入连字符;
2. 在 tabbing 环境中表示从下一行开始左边的边界移到前一个制表位, 也就是抵消一个 \+ 的效果.
- \. 产生一个重音号: \.o = ö.
- \/ 当斜体转为正体 (直立体) 时, 用此命令产生一个额外留空, 此外, 此命令也可用于避免某些西文字母间的连字 (ligature), 如 $f\,/f = ff$, 比较 $ff = ff$.
- \: [m] 在数学模式中产生一个中等长度 (2/9 quad) 的留空, 如: $xx\:x = xx\,x$.
- \; [m] 在数学模式中产生一个较大 (5/18 quad) 的留空, 如: $xx\;;x = xx\,x$.
- \< 在 tabbing 环境中出现在行的首部, 表示当前行的左边界往左移动一个制表位, 在此之前应该已用 \+ 命令使左边界往右移动了若干个制表位, 否则会出错.

- \= 1. 产生一个重音号: \=o=ô;
- 2. 在 tabbing 环境中表示在当前位置重设或添加 (不是插入) 一个制表位.
- \> 在 tabbing 环境中表示往右进到下一个制表位.
- ?‘ 产生 j.
- @ 1. 在 MakeIndex 中, 用于把 \index 命令的参数分成两部分: @ 前的部分是供条目按字母排序用的, 后面的部分则是打印输出的内容, 例如:
 $\text{\index{sum@$\sum$}}$ 表示该条目按 sum 排序, 而打印出来的是求和号 \sum ;
- 2. 在 BibT_EX 中表示条目类型, 例如 @BOOK 表示以下的条目都是 BOOK 类型的.
- \@ 插在句子末尾的大写字母与句点之间, 说明这个句点确实是句子的结束, 而不是跟在大写字母后面表示缩写词的, 提示 T_EX 在句点后面多留一点空.
- [] 为命令或环境提供可选的参数.
- \[从文本模式转换成行间数学模式的开关符, 使后面的数学公式单独成一行, 其作用相当于 \begin{displaymath}.
- \[长度] 立即换行 (不保持右端对齐), 其可选的参数长度表示在下一行之前插入一段竖直方向的空白.
- \[*[长度] 作用类似于 \[, 不过禁止在当前行与下一行之间分页.
- \] 从行间数学模式转回文本模式的开关符, 其作用相当于 \end{displaymath}.
- ~ [m] 数学式中的指数或上标, 例如: $x^2 = x^2$, $x^{-2n} = x^{-2n}$.
- \^ 产生重音号: \^o=ô.
- _ [m] 数学式中的下标, 例如: $a_n = a_n$, $a_{i,j,k} = a_{i,j,k}$.
- _ 产生下划线: t_v=t.v.
- \‘ 1. 产生重音号: \‘o=ò;
- 2. 在 tabbing 环境内, 使它后面的文本右对齐到右边界, 因此在这个命令以后不能再出现 \> 或 \=.
- { } 1. 在调用一个命令或环境时, 用来提供指定的变量;
- 2. 对文本分组, 创建一个不命名的环境;
- 3. 在 BibT_EX 中作为条目类型的文本的分隔符, 也可作为文本域的分隔符.
- \{ 产生左花括号 {.
- | [m] 产生 |.
- | 在 MakeIndex 中, 是 \index 命令内的命令字符, 相当于正常情况下的字符 ‘\’.
- 1. 在定义 \newcommand{\ii}[1]{\textit{\#1}} 后, 命令 \index{entry|ii} 的效果是在索引中把条目 ‘entry’ 的页码用 \textit 字体打印;
- 2. 在 makeidx.sty 中定义的交叉引用命令 \see 可以在 \index 内调用, 如命令

- `\index{bison|see{buffalo}}` 产生在索引内的交叉引用.
- `\|` [m] 产生 $\|$.
- `\}` 产生右花括号 `}`.
- `~` 产生单词间的正常空格, 但是禁止在这里换行, 例如 `Prof.~Jones` 保证 ‘Prof.’ 和 ‘Jones’ 位于同一行.
- `\~` 产生重音号 tilde: `\~n = ñ`.
- `\a=` 在 `tabbing` 环境内产生原本由 `\=` 定义的重音号: `\a=o = ö`.
- `\a’` 在 `tabbing` 环境内产生原本由 `\’` 定义的重音号: `\a’o = ó`.
- `\a‘` 在 `tabbing` 环境内产生原本由 `\‘` 定义的重音号: `\a‘o = ò`.
- `\AA` 产生 Å.
- `\aa` 产生 å.
- `\abovedisplayskip` [m] 一个长的行间单列公式与它前面的文本行之间的竖直间隔, 可用 `\setlength` 命令设置新值:
- `\setlength{\abovedisplayskip}{10pt plus2pt minus5pt}`
- `\abovedisplayshortskip` [m] 一个短的行间单列公式与它前面的文本行之间的竖直间隔, 同上例, 可用 `\setlength` 命令设置新值.
- `\abstractname` 含有摘要标题的命令, 在英语中定义为 ‘Abstract’, 但可被重新定义以适用于其他语言.
- `\acute{x}` [m] 用于数学变量 x 上的重音号: `\acute{a} = á`.
- `\addcontentsline{文件类型名}{格式}{条目}` 用手工方式把条目加到文件类型名所规定的目录文件 `toc`, `lof` 或 `lot`, 参数格式是指采用哪一种章节命令的标题格式, 例如:
- `\addcontentsline{toc}{section}{References}`
- `\address{发信人地址}` 用在文件类 “信件” (`letter`) 中, 以输入发信人地址, 如不止一行, 可用 `\\` 换行.
- `\addtime{秒}` 在文件类 “投影片” (`slide`) 中, 如果选取了选项 `clock`, 在 ‘注记’ (`note`) 的底部会出现时间标记, 时间标记以分为单位, 可用命令 `\settime` 设定, 本命令把指定的秒数加到时间标记中.
- `\addtocontents{文件类型名}{条目}` 用手工方式把条目加到文件类型名所规定的目录文件 `toc`, `lof` 或 `lot`, 例如:
- `\addtocontents{lof}{\protect\newpage}`
- `\addtocounter{计数器}{数}` 把一个数加到计数器中存储的当前值上.
- `\addtolength{\长度名}{长度}` 把一个长度加到长度命令 `\长度名` 的当前值上.

`\addvspace{长度}` 在本命令所在的段落后插入指定长度的竖直间隔, 此间隔被叠加到已有的间隔上, 但总和不超过长度所指定的值.

`\AE` 产生 \mathbb{A} .

`\ae` 产生 \mathbb{a} .

`\aleph [m]` 产生 \aleph .

`\allowdisplaybreaks[数] [p][a]` 本命令允许在多行的数学公式中间换页, 可选项 [数] 的值可取 0-4, 其值越大, 换页越容易发生, 如果不使用这条命令, 也可以手工换页, 只要在公式的末尾加上命令 `\displaybreak`.

`\Alph{计数器}` 用大写字母打印计数器的当前值.

`\alpha [m]` 产生 α .

`\alsiname` 用于修改宏包 `makeidx` 的命令, 在英文中, 命令 `\seealso` 的文本是 'see also', 可以重定义此命令以适合其他语言.

`\amalg [m]` 产生 \amalg .

`\and` 用在 `\author` 命令中区分作者名, 供 `\maketitle` 生成标题页.

`\angle [m]` 产生 \angle .

`\appendixname` 含有附录标题的命令, 在英语中定义为 'Appendix', 但可被重新定义以适用于其他语言.

`\approx [m]` 产生 \approx .

`\arabic{计数器}` 用阿拉伯数字打印计数器的当前值.

`\arccos [m]` 产生数学公式内的函数名 'arccos'.

`\arcsin [m]` 产生数学公式内的函数名 'arcsin'.

`\arctan [m]` 产生数学公式内的函数名 'arctan'.

`\arg [m]` 产生数学公式内的函数名 'arg'.

`\arraycolsep` 在 `array` 环境中列与列之间留空宽度的一半, 可用 `\setlength` 命令为其指定新的值:

`\setlength{\arraycolsep}{3mm}`

`\arrayrulewidth` 在 `array` 以及 `tabular` 环境中竖直或水平线的厚度, 可用命令 `\setlength` 为其指定新的长度:

`\setlength{\arrayrulewidth}{0.5mm}`

`\arraystretch` 用来改变表格内的行间距的倍数, 其正常值等于 1, 实际间距等于已定义的行间距乘以此倍数, 可用以下命令为其指定新值:

`\renewcommand{\arraystretch}{倍数}`

`\ast [m]` 产生 $*$.

`\asymp [m]` 产生 \asymp .

`\AtBeginDocument{命令序列}` [p] 先把命令序列存储起来, 等到执行命令 `\begin{document}` 时再把这段命令序列插入待处理的数据流中, 命令序列的内容可以包括只允许出现在导言中的命令, 在制作宏包时可利用这条命令以保证某些命令不会被别的宏包所覆盖.

`\AtEndDocument{命令序列}` [p] 先把命令序列存储起来, 等执行 `\end{document}` 时再把这段命令序列插入待处理的数据流中, 在制作宏包时可利用这条命令在结束文件时自动打印一些附加的内容.

`\AtEndOfClass{命令序列}` [p] 先把命令序列存储起来, 等到当前类文件输入完毕时再把这段命令序列插入待处理的数据流中. 本命令只能出现在类文件或被类文件读入的文件中, 常被用于用户的本地配置文件中, 它可能在类文件的起始部分被读入, 而此本地配置又要在类文件结束时覆盖默认的配置.

`\AtEndOfPackage{命令序列}` [p] 先把命令序列存储起来, 等到当前宏包文件输入完毕时再把这段命令序列插入待处理的数据流中. 本命令只能出现在宏包文件或被宏包文件读入的文件中, 常被用于用户的本地配置文件中, 它可能在宏包文件的起始部分被读入, 而此本地配置又要在宏包文件结束时覆盖默认的配置.

`\author{名字}` 输入作者名以供 `\maketitle` 命令生成标题页.

`\b{x}` 产生位于字母下的重音号: $\text{\b{o}} = \text{o}$.

`\backmatter` 在文件类“书籍”(book)中, 用以引入应出现在末尾的材料(如参考文献, 索引等), 本命令会关闭 `\chapter` 命令的章计数器.

`\backslash` [m] 产生 `\`.

`\bar{x}` [m] 产生数学变量上的重音号: $\text{\bar{a}} = \bar{a}$.

`\Bar{x}` [m][a] 本命令与 `\bar` 同样使用, 不过当出现 \mathcal{A} 的多重数学重音号时, 本命令会自动调节位置.

`\baselineskip` 在段落内部的行间距, 每种字体有它自己的行间距, 可以用命令 `\setlength` 为其指定新值(一个弹性长度):

`\setlength{\baselineskip}{12pt plus2pt minus1pt}`

`\baselinestretch` 正常值等于1的一个倍数, 内部长度 `\baselineskip` 乘以这个倍数后就得到实际上被使用的行间距, 可用以下命令改变它的值:

`\renewcommand{\baselinestretch}{倍数}`

不过新的值要在遇到改变字体的命令后才能生效!

`\begin{环境名}` 进入环境名指定的环境, 这个命令必须与 `\end{环境名}` 配对以退出此环境.

`\begin{abstract}` 进入“摘要”(abstract)环境以生成一个摘要, 当文件类是“文

章”(article)时,选用的字体是\small,选用的环境是“引文”(quotation),当文件类是“报告”(report)时,摘要被单独排成一页,且使用正常大小的字体与行宽,在上述两种情形里,标题Abstract都位于上部居中位置.

`\begin{align}` [a] 本命令进入行间数学模式,生成一组对齐的数学公式,以\\标志行的结束,每一行被第一、三、五、...个&号分成几列,分别对齐,如果不选用本命令的*形式,则每一行都会得到一个公式编号.

`\begin{alignat}{数}` [a] 本命令的作用类似align环境,不过它不在列对之间插入额外的空白,输入的参数等于列对的个数,也就是说,数 = $(1 + n_{\&})/2$,其中 $n_{\&}$ 是每行的&的个数,列对之间可以用手工方式插入空白,尤其当列对的左半部为空时.

`\begin{aligned}` [竖直位置] [m][a] 本命令的作用类似于align环境,不过它在数学环境里自成一个单元,可选参数竖直位置规定了它与相邻单元间的位置关系:它的取值可为t或b,分别表示按顶行或底行对齐,没有参数则表示中间对齐.

`\begin{appendix}` 进入“附录”环境以生成附录,在文件类为“文章”(book)或“报告”(report)时,章号计数器被置零,附录中相应的节号或章号以大写字母计数.

`\begin{array}[竖直位置]{列格式}` [m] 进入array环境以生成矩阵或数学模式内的阵列,列格式由每列对应一个格式字符所组成,例如`\begin{array}{lcr}`产生一个含3列的阵列,其中第一列左对齐,第二列中间对齐,第三列右对齐.可选参数竖直位置规定了这个阵列与同一行里其他单元间的位置关系:t表示顶上的行对齐,b表示底部的行对齐,没有参数则是中间对齐.参见`\begin{tabular}`.

`\begin{center}` 进入center环境,以命令\\表示行的结束,每行均居中,参见`\centering`.

`\begin{命令名}` 大多数声明性的命令,例如改变字型或字的大小的声明等,都能被作为一个环境的名称,例如`\begin{small}`与`\end{small}`之间的内容将会用\small的字体打印.

`\begin{alltt}` 在输入了alltt宏包后,此环境把除\{ }以外的字符都用打印机字体打印,包括其他特殊字符,且换行符仍起换行的作用,这样就使各种命令在此环境中仍然有效.

`\begin{bmatrix}` [m][a] 类似于matrix环境,但两边添加了方括号[].

`\begin{Bmatrix}` [m][a] 类似于matrix环境,但两边添加了花括号{}.

`\begin{cases}` [m][a] 此环境的各行以\\结束,各行均左对齐,每行中可以&号另

起一列, 并相互对齐, 最后在左边以花括号括起来, 并且竖直居中.

`\begin{description}` 进入 `description` 环境以生成一个带有缩进的罗列, 其标签的内容由命令 `\item[标签]` 规定.

`\begin{displaymath}` 从文本模式进入行间数学模式, 使数学公式单独占据一行, 其作用同 `\[`.

`\begin{document}` 进入文本文件的最外层环境, 此命令也是前言部分的结束标志, 此命令是每个 L^AT_EX 文件不可缺少的, 与其配对的是 `\end{document}`.

`\begin{enumerate}` 进入 `enumerate` 环境以生成一个带缩进以及编号的罗列, 编号数字的格式与嵌套深度有关, 最外层的编号是一个阿拉伯数字, 每遇到一个 `\item` 就增加 1.

`\begin{eqnarray}` 从文本模式进入行间数学模式, 生成按 `{rcl}` 形式分 3 列对齐的一组公式或一个多行公式, 公式的每一行均以 `\\` 结束, 行内各列以 `&` 分隔, 每个不含命令 `\nonumber` 的行都会得到一个连续的编号.

`\begin{eqnarray*}` 类似于 `eqnarray` 环境, 只是没有编号.

`\begin{equation}` 从文本模式进入行间数学模式, 生成单独占一行的数学公式, 并得到一个自动生成的编号.

`\begin{falign}` 类似于 `align` 环境, 只是在列对之间插入空白, 使它水平伸展到版面的宽度.

`\begin{figure}[位置]` 把图形嵌入文本的浮动环境, 可选参数位置可以是字母 `h`, `t`, `b`, `p` 的任意组合, 其中 `h` 表示当前位置, `t` 表示页面顶部, `b` 表示页面底部, `p` 表示单独一页, 默认值是 `tbp`, 在 L^AT_EX 2_ε 中还可以使用字母 `!` 表示取消对浮动所加的一些限制.

`\begin{figure*}[位置]` 类似于 `figure` 环境, 不过它在由可选项 `twocolumn` 或命令 `\twocolumn` 规定的双栏页面格式中占据两栏的宽度, 不像标准形式的 `figure` 只占据一栏的宽度.

`\begin{filecontents}{文件名}` [`p`] 本环境只能出现在 `\documentclass` 之前, 如果文件名所指示的文件不存在, 则创建此文件, 并把环境里的文本不作更动地复制到此文件内, 同时在文件的头部用注解的形式说明其来源, 这条命令用来传递一些非标准的附加文件, 把这些文件的内容附加在主文件的头部一起传递到别处, 在编译主文件时就能创建这些必要的附加文件, 而且可被立即引用, 如果同名文件已经存在, 则会发出一个警告信息, 并不改动已有文件.

`\begin{filecontents*}{文件名}` [`p`] 作用类似于 `filecontents` 环境, 只是不在头部附加来源信息, 因此生成新文件的内容和环境中的内容完全相同.

`\begin{flushleft}` 进入 `flushleft` 环境, 使得其中的每一行只保持左边界对齐,

不保持右边界对齐, 参见 `\raggedright`.

`\begin{flushright}` 进入 `flushright` 环境, 使得其中的每一行只保持右边界对齐, 不保持左边界对齐, 参见 `\raggedleft`.

`\begin{gather}` [a] 进入行间数学模式, 生成各行均居中的多行数学公式, 而不考虑竖直对齐, 每行均以 `\\` 结束, 而且会得到一个编号, 本命令的 `*` 形式除不给公式编号外, 其余都相同.

`\begin{gathered}` [竖直位置] [m][a] 本命令的作用类似于 `gather` 环境, 不过它在数学环境里自成一个单元, 可选参数 `竖直位置` 规定了它与相邻单元间的位置关系: 它的取值可为 `t` 或 `b`, 分别表示按顶行或底行对齐, 没有参数则表示中间对齐.

`\begin{itemize}` 进入 `itemize` 环境, 生成一个带缩进以及标签的罗列, 标签的类型与嵌套的深度有关: 最外层的标签是由 `\item` 命令产生的 `•` 号.

`\begin{letter}`{收信人} 当文件类是“信件”(letter)时创建一封信, 收信人中的内容包括收信人的姓名与地址等, 用 `\\` 表示换行.

`\begin{list}`{标准标签格式}{列表声明} 进入一般列表环境, `\item` 命令所产生的标签格式由标准标签格式规定, 而列表声明则包含一系列重定义各项列表参数的命令.

`\begin{lrbox}`{盒子名} 本命令的作用与 `\sbox` 相似, 在本命令之前应该先用命令 `\newsavebox`{盒子名} 创建一个名为 `盒子名` 的 LR 盒子, 然后本环境所包含的内容将被存储在这个盒子里, 使用命令 `\usebox`{盒子名} 即可打印此盒子的内容, 且可反复使用.

`\begin{math}` 从文本模式转为行内数学模式, 以在一行文本内插入一个数学公式, 其效果同 `(` (或文本模式中的 `$`).

`\begin{matrix}` [m][a] 其作用类似 `array` 环境, 不过列格式的声明可被忽略, 在没有列格式声明的情况下, 可以产生不超过 10 个居中的列, 否则就必须改变计数器 `MaxMatrixCols` 的值, 类似的环境还有 `pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix`, `Vmatrix`, 它们分别用 `()`, `[]`, `{ }`, `||`, `|||` 符号把阵列括起来.

`\begin{minipage}`[竖直位置][高度][内部位置]{宽度} 把文本内容排版在一个指定宽度的“小页”上, 可选项 `竖直位置` 确定了此小页相对于所处环境的位置: `t` 表示顶行对齐, `b` 表示底行对齐, 没有参数表示中间对齐, 另两个可选参数是: 高度规定总高度, 内部位置规定文本在小页内部的位置: `t` 表示顶部, `b` 表示底部, `c` 表示居中, `s` 表示扩展成充满整个空间, 默认值是 `竖直位置` 的取值, 高度中可包含参数 `\height`, `\depth`, `\width` 以及 `\totalheight`.

`\begin{multicols}`{栏数}[标题][预留高度] 这个环境是由工具包 `multicol` 提

供的, 由此处起, 页面被分成栏数所规定的几栏, 标题作为横跨的一栏打印在顶部, 如果当前页的剩余高度小于 `\premulticols` 或小于可选参数预留高度, 就会自动换页, 在末尾时是否生成新页取决于剩余高度与 `\postmulticols` 的关系, 所有页面的各栏都自动保持同样高度, 栏间距以及栏间分隔线的厚度由 `\columnsep` 与 `\columnseprule` 确定.

`\begin{multline}` [a] 进入行间数学模式以生成一个分拆成数行的数学公式, 公式的第一行向左对齐, 最后一行向右对齐, 中间各行均居中, 用 `\\` 表示换行, 利用 `\shoveleft{数学式}` 或 `\shoveright{数学式}` 可使数学式占据一行, 并向左或向右对齐. 类的选项 `reqno` (默认值) 或 `leqno` 决定了公式编号放在最后一行的右边或第一行的左边, 本环境对应的 * 形式没有公式编号.

`\begin{note}` 在文件类 “投影片” (slide) 中, 本环境生成当前投影片的一个注记, 在 L^AT_EX 2.09 中这是一个黑白片, 注记的编号是在当前投影片编号的后面用短划再加一个顺序号, 例如: 8-1, 8-2 等.

`\begin{overlay}` 在文件类 “投影片” (slide) 中, 本环境生成当前投影片的一个覆盖片, 覆盖片的编号是在当前投影片编号的后面用短划再加一个小写字母, 例如: 3-a, 3-b 等.

`\begin{picture}` (宽度, 高度) (x 坐标, y 坐标) 本环境生成一个具有指定宽度与高度的图形, 这里的 (x 坐标, y 坐标) 是图形左下角参考点的坐标, 使用的长度单位是已被定义的 `\unitlength` (默认值是 1pt).

`\begin{quotation}` 进入 `quotation` 环境, 其中的文本相对于正常边界从两边缩进, 环境内部的段落首行还有额外的缩进.

`\begin{quote}` 类似于 `quotation` 环境, 不过环境内部的段落首行没有额外的缩进, 但段落之间有额外的间距.

`\begin{slide}` 在文件类 “投影片” (slide) 中, 本环境是生成投影片的主环境.

`\begin{sloppypar}` 在这个环境里, 词与词之间的间隔允许大于通常的值, 使得一个段落可以分拆成更多的行, 参见 `\sloppy`, 与此相反的命令是 `\fussy`.

`\begin{split}` [m][a] 此环境出现在数学模式中, 其作用是使得一个公式被分拆成数行, 并用 `\\` 标注换行, 各行按 & 号竖直对齐, 其公式编号由外部环境给出, 当类选项为 `centertags` (默认值) 时, 编号竖直居中, 为 `tbtags` 时, 还要根据类选项是 `reqno` (默认值) 还是 `leqno`, 分别置放于最后一行的右边或第一行的左边.

`\begin{subarray}` {位置} {第一行 `\\` . . `\\` 最后一行} [m][a] 类似于 `\substack`, 可在数学模式里生成占据多行的上标或下标, 位置的取值可以是 `c` (居中对齐) 或 `l` (靠左对齐).

`\begin{subequations}` [a] 在这个环境里的公式编号具有一个主要数字再附加小

写字母的形式, 例如: 7a, 7b, 7c 等.

`\begin{tabbing}` 进入 `tabbing` 环境, 使得特殊的制表位命令生效: `\=` 设立一个制表位, `\>` 进到下一个制表位, `\<` 退到上一个制表位, `\\` 换行, `\+` 使左边界右移一个制表位, `\-` 使左边界左移一个制表位.

`\begin{table}[位置]` 把表格嵌入文本的浮动环境, 可选参数位置可以是字母 `h`, `t`, `b`, `p` 的任意组合, 其中 `h` 表示当前位置, `t` 表示页面顶部, `b` 表示页面底部, `p` 表示单独一页, 默认值是 `tbp`, 在 L^AT_EX 2_ε 中还可以使用字母 `!` 表示取消对浮动所加的一些限制.

`\begin{table*}[位置]` 类似于 `table` 环境, 不过它在由可选项 `twocolumn` 或命令 `\twocolumn` 规定的双栏页面格式中占据两栏的宽度, 不像标准形式的 `table` 只占据一栏的宽度.

`\begin{tabular}[竖直位置]{列格式}` 进入 `tabular` 环境以生成一个表格, 列格式由每列对应一个格式字符所组成: `c` 表示居中, `l` 表示左对齐, `r` 表示右对齐, `p{宽度}` 表示一个具有指定宽度的列, 其中的内容可占据多行, 此外, `@{文本}` 可以出现在任意两个上述的列格式之间, 其中的文本将被插入每一行的同一位置, 字符 `|` 表示贯穿各行的一条竖直线.

可选参数 `竖直位置` 规定了这个表格与外部基线间的位置关系: `t` 表示顶上的行与基线对齐, `b` 表示底部的行对齐, 没有参数则是中间对齐.

表格的各行以 `\\` 分隔, 同一行的各列则以 `&` 分隔.

`\begin{tabular*}{宽度}[竖直位置]{列格式}` 类似于 `\begin{tabular}`, 不过表格的总宽度由参数 `宽度` 给出, 为达到这个宽度必须在列的间隔中含有弹性长度, 为此可在列格式的某些地方插入 `@{\extracolsep\fill}`.

`\begin{thebibliography}{标签样本}` 进入生成参考文献的环境, 标签样本是可能出现的最长的标签, 以命令 `\bibitem` 作为一个条目的开始, 此命令为这个条目产生一个标签, 以后各行都缩进标签样本所占据的宽度.

`\begin{theindex}` 此环境以双列格式生成一个索引纪录, 每个条目的起首是命令 `\item`, `\subitem`, `\subsubitem` 再后接关键词, 或者是 `\indexspace`.

`\begin{定理环境名}[附加标题]` 此环境调用一个已被用户用命令 `\newtheorem` 定义过的定理类结构, 这里的定理环境名就是命令 `\newtheorem` 的第一个参数, 一般命名为 `theorem`, `proposition`, `axiom` 等. 附加标题 (例如: Fermat's Theorem) 则用圆括号 `()` 括起来接在定理名称及编号的后面.

`\begin{titlepage}` 此环境产生一个没有页码的标题页, 用户对此页面的构成有完全的控制.

`\begin{trivlist}` 此环境生成一个既无标签样本又无格式声明的平凡列表, 参

数 `\leftmargin`, `\labelwidth`, `\itemsep` 都被置为 0pt, 而 `\listparindent = \parindent`, `\parsep = \parskip`.

`\begin{verbatim}` 此环境内的文本将被一字不差地按输入的样式打印出来, 包括空白行、换行、注解等全都原原本本被打印. (注意: 其中不能含汉字)

`\begin{verbatim*}` 类似于 `verbatim` 环境, 只是文中的空格被打印成 `\`.

`\begin{verse}` 用于写诗歌、韵文等的环境, 用 `\\` 提示换行, 而段与段用空白行分隔.

`\begin{vmatrix}` [m][a] 类似于 `matrix` 环境, 不过两端用 `||` 括起来.

`\begin{Vmatrix}` [m][a] 类似于 `matrix` 环境, 不过两端用 `|||` 括起来.

`\belowdisplayskip` [m] 一个长的单列公式与后面的文本行之间的竖直留空, 可用 `\setlength` 为其指定新值, 例如: 命令

`\setlength{\belowdisplayskip}{\abovedisplayskip}`

使 `\belowdisplayskip` 取与 `\abovedisplayskip` 相同的值, 参见 `\abovedisplayskip`.

`\belowdisplayshortskip` [m] 一个短的单列公式与后面的文本行之间的竖直留空, 可用 `\setlength` 为其指定新值 (参见上一条目).

`\beta` [m] 产生 β .

`\bezier{点数}(x_1, y_1)(x_2, y_2)(x_3, y_3)` 本命令应被用于 `picture` 环境内, 生成一条端点为 (x_1, y_1) , (x_3, y_3) 的二次 Bézier 曲线, 以 (x_2, y_2) 作为控制点, 此曲线由 (点数 + 1) 个点构成, 本命令类似于 `\qbezier`, 不过后者的点数是自动产生的.

`\bf` 转变为罗马族, 直立形状, 黑体系列 (**Roman, upright, bold**) 的字体属性.

`\bfdefault` 定义了由命令 `\bfseries` 选取的字体系列, 可用 `\renewcommand` 予以重定义:

`\renewcommand{\bfdefault}{b}`

`\bfseries` 本声明不改变当前字体的族与形状, 但转变成 **bold** 序列.

`\bibitem[标签]{引用词}` 条目文本 本命令应被用于 `thebibliography` 环境内, 生成一个参考文献条目, 当正文中引用这个文献时, 以引用词作为命令 `\cite` 的参数, 此时 `\cite` 命令的位置将被带方括号的可选项 [标签] 所取代, 在默认的情形则被一个带方括号的 [标号] 所取代.

`\bibliography{文件名}` 利用程序 `BibTEX` 生成参考文献: 文件名是包含被搜索的文献数据库的一个或多个文件的根文件名.

`\bibliographystyle{格式}` 在与程序 `BibTEX` 配合使用时, 本命令选取文献条目的打印格式, 格式的选取可以是 `plain`, `unsrt`, `alpha`, `abbrv`, 默认的是第

一个,也可以有其他非标准的格式.

`\bibname` 在文件类为“书籍”(book)或“报告”(report)时,本命令含有参考文献的标题,在英文中是‘Bibliography’,可根据不同的语言加以修改.

`\big`定界符 [m] 产生比正常大但比`\Big`小的定界符,例如: `\big(=` (.

`\Big`定界符 [m] 产生比`\big`大但比`\bigg`小的定界符,例如: `\Big[=` [.

`\bigcap` [m] 产生 \cap .

`\bigcirc` [m] 产生 \bigcirc .

`\bigcup` [m] 产生 \cup .

`\bigg`定界符 [m] 产生比`\Big`大但比`\Bigg`小的定界符,例如: `\bigg| =` |.

`\Bigg`定界符 [m] 最大的定界符,例如: `\Bigg\langle =` $\left\langle$.

`\biggl`定界符 [m] 类似于`\bigg`,但它也是左定界符.

`\Biggl`定界符 [m] 类似于`\Bigg`,但它也是左定界符.

`\biggm`定界符 [m] 类似于`\bigg`,但它与两边的符号有更大的留空(作为关系算子).

`\Biggm`定界符 [m] 类似于`\Bigg`,但它与两边的符号有更大的留空(作为关系算子).

`\biggr`定界符 [m] 类似于`\bigg`,但它也是右定界符.

`\Biggr`定界符 [m] 类似于`\Bigg`,但它也是右定界符.

`\bigl`定界符 [m] 类似于`\big`,但它也是左定界符.

`\Bigl`定界符 [m] 类似于`\Big`,但它也是左定界符.

`\bigm`定界符 [m] 类似于`\big`,但它与两边的符号有更大的留空(作为关系算子).

`\Bigm`定界符 [m] 类似于`\Big`,但它与两边的符号有更大的留空(作为关系算子).

`\bigodot` [m] 产生 \odot .

`\bigoplus` [m] 产生 \oplus .

`\bigotimes` [m] 产生 \otimes .

`\bigr`定界符 [m] 类似于`\big`,但它也是右定界符.

`\Bigr`定界符 [m] 类似于`\Big`,但它也是右定界符.

`\bigtriangledown` [m] 产生 ∇ .

`\bigtriangleup` [m] 产生 \triangle .

`\bigskip` 插入一个值为`\bigskipamount`的大的竖直间隔,参见`\medskip`以及`\smallskip`.

`\bigskipamount` 用于`\bigskip`的竖直间隔的标准值,可用命令`\setlength`加以改变:

- `\setlength{\bigskipamount}{6ex plus1.5ex minus2ex}`
- `\bigsqcup [m]` 产生 \sqcup .
- `\biguplus [m]` 产生 \uplus .
- `\bigvee [m]` 产生 \vee .
- `\bigwedge [m]` 产生 \wedge .
- `\binom{上部}{下部} [m][a]` 产生组合数: $\binom{n}{k}$.
- `\bmod [m]` 产生函数名 'mod', 例如: $a\bmod b = a \bmod b$.
- `\boldmath` 转移到数学模式的黑体字, 请注意: 本命令必须在进入数学模式前出现在文本模式中, 如果只需把公式的一部分转成黑体, 则也可使用命令 `\mbox{\boldmath$...$}` 以暂时进入文本模式.
- `\boldsymbol{符号} [m][a]` 当宏包 `amsmath` 或 `amsbsy` 之一被读入后, 本命令可把符号打印成黑体, 与 `\mathbf` 不同, 本命令对数学符号以及小写希腊字母也有效.
- `\bot [m]` 产生 \bot .
- `\botfigrule` 本命令仅当一个浮动环境出现在页底时被执行, 它通常被定义为空, 但可被重定义以在页面的正文与浮动部分之间画一条直线, 请注意这条命令不应该产生额外的竖直空间, 例如:
- ```
\renewcommand{\botfigrule}{\vspace*{-.4pt}
\rule{\columnwidth}{.4pt}}
```
- `\bottomfraction` 页面中可供底部的浮动单元使用的最大部分, 这是一个十进小数, 可用以下命令重置其值:
- ```
\renewcommand{\bottomfraction}{0.5}
```
- `bottomnumber` 此计数器记录了可出现在一个页面底部的浮动单元最大个数, 可用命令 `\setcounter{bottomnumber}{3}` 为其重置新值.
- `\bowtie [m]` 产生 \bowtie .
- `\Box [m]` 产生 \Box .
- `\Boxed{公式} [m][a]` 把公式框起来.
- `\breve{x} [m]` 产生数学变量 x 的重音号: $\breve{a} = \check{a}$.
- `\Breve{x} [m][a]` 类似于 `\breve`, 但当遇到多重 $\mathscr{}$ -L^AT_EX 重音号时, 会自动调整其位置.
- `\bullet [m]` 产生 \bullet .
- `\c{x}` 在 x 的底下产生一个变音号 cedilla: $\c{C} = \mathring{C}$.
- `\cal [m]` 是 L^AT_EX 2.09 中使用的声明, 在数学模式内选取书写体 (calligraphic), 在 L^AT_EX 2_ε 中已被 `\mathcal` 取代.

`\cap [m]` 产生 \cap .

`\caption[简短形式]{标题文本}` 在浮动环境 `figure` 或 `table` 内生成一个含有编号以及标题文本的标题, 可选项简短形式是用来在目录中代替标题文本的.

`\captions语言` 如果输入了宏包 `esperant`, `german` 或者 `babel`, 此命令用于在多语言环境中重定义一些特定的标题, 例如 ‘Chapter’, ‘Contents’ 等, 本命令通常是命令 `\selectlanguage` 定义的一部分.

`\cc{表}` 在文件类 “信件” (`letter`) 中, 本命令在信末生成 ‘cc:’, 后接一个姓名的表.

`\ccname` 在文件类 “信件” (`letter`) 中, 本命令含有 `\cc` 命令所打印出来的文字, 在英文环境里是 ‘cc’, 但可根据使用的语言重新定义.

`\cdot [m]` 产生 \cdot .

`\cdots [m]` 产生 \cdots .

`\centering` 转换成每行皆居中的格式, 用 `\\` 标志换行, 参见 `\begin{center}`.

`\centerline{文本}` 让文本居中成一行的 T_EX 命令.

`\cfrac[位置]{上部}{下部} [m][a]` 本命令产生一个连分数, 可选项位置可取值 1 或 r, 分别表示分子向左或右对齐, 默认是居中对齐.

`\chapter[短名]{章名}` 另起一页开始新的一章, 以一个自动产生的编号加上章名作为章的标题, 如果给出了可选项短名, 那么它将在目录以及书眉上取代原有的章名.

`\chapter*{章名}` 另起一页开始新的一章, 以章名作为章的标题, 但没有编号, 而且章名也不出现在目录中.

`\chaptername` 本命令包含章的标题, 在英文环境中是 ‘Chapter’, 可被重新定义以适应其他语言.

`\check{x} [m]` 在数学变量 x 上产生重音号: `\check{a} = ä`.

`\Check{x} [m][a]` 类似于 `\check`, 但是在出现多重 \mathcal{A} \mathcal{M} \mathcal{S} -L^AT_EX 重音号时, 会自动调整位置.

`\CheckCommand{\命令名}[变量数][可选参数]{定义}` 检查命令 `\命令名` 的当前定义是否与料想的定义一致, 若不一致, 就会发出一个出错信息, 此命令用于保证重要的命令不会被别的宏包修改.

`\CheckCommand*{\命令名}[变量数][可选参数]{定义}` 类似于 `\CheckCommand`, 但是要求被检查的命令是短的命令, 即在定义中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\chi [m]` 产生 χ .

`\circ [m]` 产生 \circ .

`\circle{直径}` 在图形 (picture) 环境里产生一个指定直径的圆周, 本命令应被用在命令 `\put` 或 `\multiput` 中作为参数.

`\circle*{直径}` 类似于 `circle`, 不过产生一个实心圆.

`\cite[附注]{引用词}` 利用引用词在正文中产生对参考文献标签的引用, 可选参数附注被添加在标签的后面.

`\ClassError{类名}{出错信息}{帮助}` [p] 本命令只能出现在类文件中, 它向监视器以及纪录文件发出以类名为标号的出错信息, 中断进程并等待用户的反应, 如果用户键入 H<回车>, 就把帮助打印出来, 在出错信息和帮助中都可用 `\MessageBreak` 命令换行, 用 `\space` 强制产生空格, 并可用 `\protect` 冠在一个命令前面, 使得这个命令不被解释执行而是把它的名字打印出来.

`\ClassInfo{类名}{信息}` [p] 类似于 `\ClassWarningNoLine`, 不过信息不在监视器上显示, 只是写入纪录文件.

`\ClassWarning{类名}{警告信息}` [p] 本命令只能出现在类文件中, 它向监视器以及纪录文件发出以类名为标号的警告信息, 同时给出输入文件的当前行号, 并继续进行下去, 警告信息可按 `\ClassError` 同样的方法格式化.

`\ClassWarningNoLine{类名}{警告信息}` [p] 类似于 `\ClassWarning`, 只是不打印当前行号.

`\cleardoublepage` 结束当前页, 并把所有尚未处理的浮动单元输出到一个或几个浮动页上, 接在后面的页面将是具有奇数页码的右页.

`\clearpage` 结束当前页, 并把所有尚未处理的浮动单元输出到一个或几个浮动页上.

`\cline{n-m}` 在 `tabular` 环境里生成一条从第 n 列到第 m 列的水平线, 例如:
`\cline{2-5}`.

`\closing{问候语}` 信件 (letter) 环境里正文结束后, 问候语用于收尾部分.

`\clubsuit` [m] 产生 ♣.

`\color` 色彩 必须先输入宏包 `color` 才能使用本命令, 它是选定某种色彩的声明, 以后的文本就用这种颜色打印, 它的作用范围持续到所在环境的结束或者遇到另一条 `\color` 命令为止. 色彩可以是预定义的或已用 `\definecolor` 定义过的颜色名, 也可具有 [模式]{数据} 的形式, 其中参数数据的意义同 `\definecolor`, 例如:

`\color[rgb]{0.5,0.5,0}` `\color{magenta}`

`\colorbox` 色彩{文本} 必须先输入宏包 `color` 才能使用本命令, 文本生成一个 LR 盒子, 并以指定的色彩作为盒子的背景色, 色彩的定义同 `\color`.

`\columnsep` 规定了双栏页面格式中的栏间距值, 可用 `\setlength` 重置:

`\setlength{\columnsep}{1pt}`

`\columnseprule` 规定了双栏页面格式中的栏间竖直线的厚度, 可用 `\setlength` 重置: `\setlength{\columnseprule}{1pt}`

`\cong [m]` 产生 \cong .

`\contentsline{章节类型}{\numberline{编号}标题}{页码}` 本命令出现在 `toc` 文件内, 每一个命令对应目录中的一个条目, 当 T_EX 遇到 `\tablecontents` 命令时, 才把这个文件读入, 因此用户可根据需要利用文本编辑器修改或添加 `toc` 文件中的这种命令. 章节类型是指章节的层次, 例如 `section`, 编号是此章节的编号 (例如 2.3), 页码是正文中出现的页码.

`\contentsname` 本命令包含目录的标题, 在英文环境里是 'Contents', 可根据需要重新定义以与所使用的语言一致.

`\coprod [m]` 产生 \coprod .

`\copyright` 产生 ©.

`\cos [m]` 本命令生成数学公式里的函数名 'cos'.

`\cosh [m]` 本命令生成数学公式里的函数名 'cosh'.

`\cot [m]` 本命令生成数学公式里的函数名 'cot'.

`\coth [m]` 本命令生成数学公式里的函数名 'coth'.

`\csc [m]` 本命令生成数学公式里的函数名 'csc'.

`\cup [m]` 产生 \cup .

`\CurrentOption [p]` 本命令包含当前正被处理的可选项名称, 它仅能用于可选项的定义中, 特别是默认可选项的定义中.

`\d{x}` 产生置于字母底下的一点: `\d{o} = ọ`.

`\dag` 产生 †.

`\dagger [m]` 产生 †.

`\dashbox{虚线长}{宽度, 高度}[位置]{文本}` 本命令仅能用于 `picture` 环境中, 生成一个带虚线框的盒子, 具有指定的宽度、高度以及虚线长, 可选参数位置是指文本在盒子内部的位置, 它们可以是居左 (l), 居右 (r), 居顶 (t) 或居底 (b), 也可以是两者的组合, 如 `lt`, 默认值是居中, 本命令被用作 `\put` 或 `\multiuput` 命令的参数.

`\dashv [m]` 产生 \dashv .

`\date{日期}` 1. `\maketitle` 命令通常在目录页中打印当前日期, 用户可利用这条命令打印任何文本.

2. 在“信件”内打印指定的日期以取代自动打印当前日期.

`\date`语言 如果输入了宏包 `esperant`, `german` 或者 `babel`, 此命令用于在多语言环境中重定义 `\today` 命令, 例如 `\dateUSenglish`, `\dateenglish` 等, 本命令通常是命令 `\selectlanguage` 定义的一部分.

`\dbinom{上部}{下部} [m][a]` 类似于 `\binom`, 生成一个组合数, 不过其中字母的尺寸是 `\displaystyle`.

`\dblfigrule` 本命令仅当在一个双栏的浮动环境出现在页顶时被执行, 它通常被定义为空, 但可被重定义以在页面的正文与浮动部分之间画一条直线, 请注意这条命令不应该产生额外的竖直空间, 例如:

```
\renewcommand{\dblfigrule}{\vspace*{-.4pt}
\rule{\textwidth}{.4pt}}
```

`\dblfloatpagefraction` 在双栏页面格式中, 一个浮动页至少必须占据一页的多少部分才能再生成一个新的页, 这是一个十进小数, 可用以下命令指定它的新值:

```
\renewcommand{\dblfloatpagefraction}{0.8}
```

`\dblfloatsep` 在双栏页面格式中, 上下两个跨栏的浮动单元间的竖直间隔, 可用以下命令指定它的新值:

```
\setlength{\dblfloatsep}{12pt plus2pt minus4pt}
```

`\dbltextfloatsep` 在双栏页面格式中, 一个位于页顶的浮动单元与在它下面的正文间的竖直间隔, 可用 `\setlength` 命令指定它的新值.

`\dbltopfraction` 在双栏页面格式中, 一个页面可被位于顶部的跨栏浮动单元占据的最大部分, 这是一个十进小数, 可用以下命令指定它的新值:

```
\renewcommand{\dbltopfraction}{0.3}
```

`dbltopnumber` 在双栏页面格式中, 一个页面顶部可出现的跨栏浮动单元的最大个数, 可用以下命令指定它的新值:

```
\setcounter{dbltopnumber}{3}
```

`\ddag` 产生 ‡.

`\ddagger [m]` 产生 ‡.

`\ddot{x} [m]` 在数学公式中产生一个两点重音号: `\ddot{a}=ä`.

`\DeclareErrorFont{编码}{族}{系列}{形状}{尺寸} [p]` 如果找不到有效的字体, 甚至连 `\DeclareFontSubstitution` 所给出的替代字体也找不到, 那么本命令声明的字体是最后的救星, 例如:

```
\DeclareErrorFont{OT1}{cmr}{m}{n}{10}
```

`\DeclareFixedFont{命令名}{编码}{族}{系列}{形状}{尺寸} [p]` 把 `\命令名` 定义为选取这条命令所规定的字体的声明.

`\DeclareFontEncoding{编码}{文本模式命令}{数学模式命令}` [p] 声明创立以编码命名的字体编码体系, 文本模式命令是转换成文本模式时需要执行的命令序列, 同理, 数学模式命令是转换成数学模式时需要执行的命令序列, 编码有: OT1 (Knuth 创建的文本字体, 如 cmr10), OT2 (华盛顿大学的斯拉夫字体, 如 wncyr10), T1 (DC 字体, 如 dcr10), OML (T_EX 数学字母字体, 如 cmml10), OMS (T_EX 数学符号字体, 如 cmsy10), OMX (T_EX 数学扩充字体, 如 cmex10), U (未知编码), 例如:

```
\DeclareFontEncoding{OT1}{}{}
```

`\DeclareFontEncodingDefaults{文本模式命令}{数学模式命令}` [p] 声明对所有字体编码体系都适用的进入文本或数学模式时需要执行的命令序列, 具体编码体系的附加命令序列都接在后面执行.

`\DeclareFontFamily{编码}{族名}{命令序列}` [p] 声明创立以族名命名的新字体族, 以已有定义的编码作为编码体系, 每当这个字体族被选取时, 就要先执行指定的命令序列.

`\DeclareFontShape{编码}{族}{系列}{形状}{定义}{命令序列}` [p] 本命令把外部的字库文件名与指定的字体属性相关连, 在定义中把字体尺寸与字库名相联系, 每当这种字体被选用时, 就要先执行指定的命令序列.

`\DeclareFontSubstitution{编码}{族}{系列}{形状}` [p] 如果对应某种字体属性的有效字体不存在, 就用本命令指定的字体属性来替代, 替代次序是: 形状、系列、族, 编码不能替代. 例如:

```
\DeclareFontSubstitution{U}{cmr}{m}{n}
```

`\DeclareGraphicsExtensions{扩展名表}` [p] 建立可被 `\includegraphics` 命令以及 `graphics`, `graphicx` 宏包输入的图形文件的默认扩展名表, 扩展名表用逗号分隔, 如 `eps`, `ps` 等.

`\DeclareGraphicsRule{扩展名}{类型}{信息文件扩展名}{命令}` [p] 把图形文件的扩展名与图形文件的类型、含有图框尺寸的信息文件扩展名以及一个操作命令相关连, 执行此命令后图形方能被输入, 例如:

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{'gunzip -c #1}
```

其中的命令必须以 ' 为前缀, #1 代表被处理的文件名.

`\DeclareMathAccent{命令}{类型}{符号字体}{编号}` [p] 声明 `\命令` 为数学重音命令, 打印内部名称为符号字体中的具有指定编号的字符, 类型可以是 `\mathord` 或 `\mathalpha`, 在后一情形, 符号随着数学字体的变化而变化, 例如:

```
\DeclareMathAccent{acute}{\mathalpha}{operators}{19}
```

`\DeclareMathAlphabet{命令}{编码}{族}{系列}{形状}` [p] 把 `\命令` 定义为数

学模式下设置字体的命令, 对所有的数学变体 (除了正常字体 (normal) 外, 目前仅有的变体是 bold, 用 `\boldmath` 选取), 选取的是同样的具有指定属性的字体, 如果想对不同的变体使用不同字体的话, 还需要用 `\SetMathAlphabet` 个别地定义. 当形状属性留空时, 本命令虽然被创立了, 但对所有的变体都是未经定义的, 仍需要用 `\SetMathAlphabet` 对各种变体分别予以定义, 例如:

```
\DeclareMathAlphabet{\mathsl}{OT1}{cmr}{m}{sl}
```

`\DeclareMathDelimiter{\命令}{类型}{字体一}{编号}{字体二}{编号}` [p] 声明 `\命令` 是有两种不同大小的定界符, 小的变体取自内部名称为字体一的具有所给编号的字符, 大的变体取自内部名称为字体二的具有所给编号的字符, 例如:

```
\DeclareMathDelimiter{()}{\mathopen}{operators}{40}
{\largesymbols}{0}
```

`\DeclareMathOperator{\命令}{函数名}` [p][a] 在输入宏包 `amsopn` 与 `amsmath` 后, `\命令` 定义为数学模式里的命令, 它用直立字体打印函数名, 并有适当的留空, 相应的 * 形式命令还可以用 `~` 与 `_` 输入上下界.

`\DeclareMathRadical{\命令}{字体一}{编号}{字体二}{编号}` [p] 声明 `\命令` 是有两种不同大小的数学根式符号, 小的变体取自内部名称为字体一的具有所给编号的字符, 大的变体取自内部名称为字体二的具有所给编号的字符, 例如:

```
\DeclareMathRadical{\sqrtsign}{symbols}{112}{largesymbols}{112}
```

`\DeclareMathSizes{正文}{数学正文}{角标}{二级角标}` [p] 规定 3 种数学字体 `\textstyle`, `\scriptstyle`, `\scriptscriptstyle` 的大小点数, 这里 4 个参数都是整数, 其单位是 pt, 第一个参数正文是正文字体的点数, 后面 3 个分别对应上述 3 种数学字体的点数, 例如:

```
\DeclareMathSizes{10}{10}{7}{5}
```

`\DeclareMathSymbol{\命令}{类型}{字体}{编号}` [p] 声明 `\命令` 是一个数学符号, 打印出内部名称为字体的具有所给编号的字符, 它在所有数学字体命令下打印同一个符号, 但对于不同的变体可能有不同的符号字体, 不过这必须由 `\Set...` 命令设置成别的属性, 这里的类型可以是: `\mathord` (通常的符号), `\mathop` (大型算子, 如 \sum), `\mathbin` (二元算子, 如 \times), `\mathrel` (关系算子, 如 \geq), `\mathopen` (开括号, 如 $($), `\mathclose` (闭括号, 如 $)$), `\mathpunct` (标点符号), `\mathalpha` (字母), 例如:

```
\DeclareMathSymbol{\alpha}{\mathord}{letters}{11}
```

`\DeclareMathVersion{变体名}` [p] 声明数学字体或符号的一个新变体, 开始时此变体将使用 `\Declare..` 命令所规定的字体属性, 但它可以被合适的 `\Set..` 命令重新设置, 在正文中可用 `\mathversion{变体名}` 命令选取此变体.

`\DeclareOldFontCommand{\命令}{文本模式命令}{数学模式命令}` [p] 把 `\命令` 定义为字体声明, 文本模式时使用文本模式命令, 数学模式时使用数学模式命令, 本命令的目的是定义与 L^AT_EX 相容的命令, 一般应避免使用, 例:

```
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
```

`\DeclareOption{选项}{命令序列}` [p] 在类文件或宏包中, 本命令给出了与选项相关的命令序列, 这些命令是在调用 `\ExecuteOptions` 或 `\ProcessOptions` 时被执行的, 后两条命令执行后, 所有的定义都从内存中清除, 命令序列则被存储在内部命令 `ds@选项` 中.

`\DeclareOption*{命令序列}` [p] 在类文件或宏包中, 给出了与所有未定义的选项相关联的默认命令序列, 命令序列中可使用的特殊命令是 `\CurrentOption` (指选项名) 及 `\OptionNotUsed`, 例如:

```
\DeclareOption*{\InputIfFileExists
  {\CurrentOption.sty}{\OptionNotUsed}}
```

`\DeclareRobustCommand{\命令}[变量数][选项]{定义}` 类似于 `\newcommand` 那样定义或重定义 `\命令`, 只是所得的命令是坚固的: 它可用作其他命令的变量, 而不需在前面冠以 `\protect`,

`\DeclareRobustCommand*{\命令}[变量数][选项]{定义}` 类似于不带星的命令 `\DeclareRobustCommand`, 只是定义必须是短的: 即在定义中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\DeclareSymbolFont{字体}{编码}{族}{系列}{形状}` [p] 声明具有指定字体属性的字体为符号字体, 其内部名称为第一个参数字体, 此符号字体可用于所有的数学变体, 否则要用 `\SetSymbolFont` 重定义.

`\DeclareSymbolFontAlphabet{\命令}{字体}` [p] 把 `\DeclareSymbolFont` 定义的内部名称为字体的字体声明为 `\命令` 所指的数学字体, 当已经定义了一个合用的数学字体时, 此命令比 `\DeclareMathAlphabet` 更适用, 例如:

```
\DeclareSymbolFontAlphabet{\mathrm}{operators}
```

`\DeclareTextAccent{\命令}{编码}{字符编号}` [p] 当指定的编码处于激活状态时, `\命令` 定义为重音号命令, 它以字体中位于字符编号位置的符号作为重音号, 例如:

```
\DeclareTextAccent{\'}{OT1}{19}
```

`\DeclareTextAccentDefault{\命令}{编码}` [p] 当缺乏激活的编码时, 本命令声明用于重音号命令的指定编码.

`\DeclareTextCommand{\命令}{编码}[变量数][选项]{定义}` [p] 像 `\newcommand` 那样定义 `\命令`, 只是仅当指定的编码处于激活状态时, 定义才有效.

`\DeclareTextCommandDefault{\命令}[变量数][选项]{定义}` [p] 对所有的编码都为`\命令`建立一个默认的定义.

`\DeclareTextComposite{\命令}{编码}{字母}{字符编号}` [p] 当指定的编码处于激活状态时, 把`\命令`后接字母定义为字体中位于字符编号位置的字符, 于是在 T1 编码, 并且带有重音号的字母有一个单独的字符与之对应时, 就定义了重音号命令的一个作用, 例如: ♦

```
\DeclareTextComposite{\'}{T1}{e}{233}
```

当然这个重音号命令必须用 `\DeclareTextAccent` 或 `\DeclareTextCommand` (带一个参数) 对于指定的编码定义过.

`\DeclareTextCompositeCommand{\命令}{编码}{字母}{定义}` [p] 类似于上述命令 `\DeclareTextComposite`, 只是对于`\命令`后接字母的组合可以指定任意的定义.

`\DeclareTextFontCommand{\命令}{定义}` [p] 定义`\命令`为指定文本字体的命令, 例如:

```
\DeclareTextFontCommand{\textbf}{\bfseries}
```

`\DeclareTextSymbol{\命令}{编码}{字符编号}` [p] 当指定的编码处于激活状态时, 定义`\命令`为打印位于字符编号位置的字符, 例如:

```
\DeclareTextSymbol{\AE}{OT1}{29}
```

`\DeclareTextSymbolDefault{\命令}{编码}` [p] 当缺乏激活的编码时, 本命令声明用于符号命令的指定编码.

`\definecolor{色彩名}{模式}{数据}` 必须先输入宏包 `color` 才有效, 它给色彩名指定一种颜色, 可用的模式为: `rgb` (红, 绿, 蓝), `cmYk` (青, 洋红, 黄, 黑), `gray` (灰) 以及 `named` (已命名的). 数据是一组用逗号分隔的 0 与 1 之间的十进小数串, 表示每个分量的强度. 在 `named` 模式, 数据应取成驱动程序认识的色彩名, 例如:

```
\definecolor{litegrn}{cmYk}{0.25,0,0.75,0}
```

```
\definecolor{brown}{named}{RawSienna}
```

在所有的彩色驱动程序中都被预定义的颜色有: `red`, `green`, `blue`, `yellow`, `cyan`, `magenta`, `black`, `white`.

`\deg` [m] 在数学公式中产生函数名 'deg' 的命令.

`\DeleteShortVerb{\字符}` 当标准宏包 `shortvrb` 被输入后, 本命令取消前面的命令 `\MakeShortVerb{\字符}` 的作用, 使字符恢复其本来的意义.

`\Delta` [m] 产生 Δ .

`\delta` [m] 产生 δ .

`\depth` 表示盒子的深度 (从基线到底部) 的长度参数, 它只能用于指定 `\makebox`, `\framebox` 或 `\savebox` 的宽度参数中, 或者用于 `\parbox` 及 `minipage` 环境的高度参数中, 例如:

```
\framebox[20\depth]{text}
```

`\det [m]` 在数学公式中产生函数名 ‘det’ 的命令, 可带下标作为下界.

`\dfrac{分子}{分母} [m][a]` 类似 `\frac` 产生一个分式, 不过是以 `\displaystyle` 的大小.

`\DH` 当 T1 编码被激活时, 产生字母 Đ.

`\dh` 当 T1 编码被激活时, 产生字母 đ.

`\Diamond [m]` 产生 ◇.

`\diamond [m]` 产生 ◇.

`\diamondsuit [m]` 产生 ◇.

`\dim [m]` 在数学公式中产生函数名 ‘dim’ 的命令.

`\discretionary{前面}{后面}{整体}` 单词内部的可能拆分方案, 它可被分成两部分, 使前面在行尾, 后面在下一行的头, 如不被拆开时则打印整体, 例如在德文中可能有:

```
\discretionary{k-}{k}{ck}
```

`\displaybreak[数] [m][a]` 本命令允许在多行数学公式里换页, 只要插在 `\\` 之前, 数可以取 0-4, 按递增次序鼓励换页, 如想在多行公式内自动换页, 则必须在前言部分发出 `\allowdisplaybreaks` 命令.

`\displaystyle [m]` 在数学模式里使字体大小按 `\displaystyle` 选取.

`\div [m]` 产生 ÷.

`\DJ` 当 T1 编码被激活时, 产生字母 Đ.

`\dj` 当 T1 编码被激活时, 产生字母 đ.

`\documentclass[选项]{类}[版本号] [p]` 通常是 L^AT_EX 2_ε 文件的第一个命令, 它确定了文件的整体特征, 类的标准取值为

article, report, book, letter, slides

一次只能选取其中之一, 选项可在下表中选取, 多于一个时可用逗号分隔:

10pt, 11pt, 12pt,

letterpaper, legalpaper, executivepaper,

a4paper, a5paper, b5paper, landscape,

onecolumn, twocolumn,

oneside, twoside,

notitlepage, titlepage,

leqno, fleqn, openbib,

draft, final

这些或其他增加的选项都是整体的, 它们对后面由 \usepackage 命令输入的宏包都有效,

版本号是形如 yyyy/mm/dd 的一个日期, 例如: 1994/08/01, 如果输入类文件的日期比它早, 就会打印一个警告信息.

\documentstyle[选项]{格式} [p] 通常是 L^AT_EX 2.09 文件的第一个命令, 它确定了文件的整体特征, 在 L^AT_EX 2_ε 里已被 \documentclass 取代, 不过它仍可进入兼容模式, 以处理老式文件.

\dot{x} [m] 在数学公式里打印重音号: \dot{a} = \dot{a} .

\Dot{x} [m][a] 类似于 \dot, 不过遇到多重 L^AT_EX 重音号时会自动调整位置.

\doteq [m] 产生 \doteq .

\dotfill 用点线填满一段空间, 例如: = \dotfill.

\dots 产生

\dots [m][a] 在数学模式里打印 3 个点, 其竖直位置会根据后继符号自动确定.

\dotsb [m][a] 接在二元算子后的 \dots:

\dotsc [m][a] 接在逗号后的 \dots:

\dotsi [m][a] 接在积分号后的 \dots:

\dotsm [m][a] 当乘号用的 \dots, 同 \dotsb:

\Downarrow [m] 产生 \Downarrow .

\downarrow [m] 产生 \downarrow .

\ell [m] 产生 ℓ .

\em 声明选用强调字体, 它与当前字体同族、同系列, 但形状不同, 通常是在直立字体与斜体间转换.

\emph{文本} 用强调字体打印文本, 它与当前字体同族、同系列, 但形状不同, 通常是在直立字体与斜体间转换, 此命令等价于 {\em 文本 /}, 也就是说, 自动插入了斜体附加留空.

\emptyset [m] 产生 \emptyset .

\encl{附件表} 文件类 “信件” (letter) 中的命令, 在信末打印 ‘encl.’ 再接上附件表, 词 ‘encl’ 包含在命令 \enclname 中, 可根据需要重新定义.

\enclname 文件类 “信件” (letter) 中的命令, 含有命令 \encl 所打印的词, 在英文环境中是 ‘encl’, 可根据需要重新定义.

\end{环境名} 结束由命令 \begin{环境名} 进入的环境.

`\enlargethispage{长度}` 使 `\textheight` 暂时增加长度, 以避免出现一个难看的页面, 换页后, `\textheight` 将恢复原值.

`\enlargethispage*{长度}` 类似于 `\enlargethispage`, 不过同时取消所有附加的行间留空, 使这一页能容纳更多内容.

`\ensuremath{数学命令}` 不论在文本模式还是数学模式中使用本命令, 都能使数学命令在数学模式中执行, 这个命令主要用于定义需要数学模式的新命令, 使其在文本模式中也能调用.

`\epsilon [m]` 产生 ϵ .

`\eqref{标记} [a]` 这是 `\ref` 命令的变体, 打印出带括号的由命令 `\label{标记}` 定义的公式编号, 例如: (6.5).

`\equiv [m]` 产生 \equiv .

`\eta [m]` 产生 η .

`\evensidemargin` 偶数页的左页边宽, 仅在文件类“书籍”(book), 或其他文件类但选项 `twoside` 被选取时才有效, 可用 `\setlength` 命令重置新值, 例如:

```
\setlength{\evensidemargin}{2.5cm}
```

`\ExecuteOptions{选项表} [p]` 在一个类文件或宏包里, 本命令将执行选项表里的所有选项的定义, 通常在命令 `\ProcessOptions` 之前调用本命令, 以使某些选项成为默认的.

`\exists [m]` 产生 \exists .

`\exp [m]` 在数学公式中产生函数名 ‘exp’ 的命令.

`\extracolsep{附加宽度}` 用于 `tabular` 列格式命令中, 在以后各列之间产生附加宽度的间隔, 本命令用在列格式的 `@` 表达式中:

```
\begin{tabular}{lr@{\extracolsep{2.5mm}}lcr}
```

`\fbox{文本}` 产生一个边框: `\fbox{text} =`

text

`.`

`\fboxrule` 由 `\fbox` 或 `\framebox` 产生的边框线条厚度, 可用 `\setlength` 为其重置新值:

```
\setlength{\fboxrule}{1pt}
```

`\fboxsep` 由 `\fbox` 或 `\framebox` 产生的边框与文本的间隔, 可用 `\setlength` 为其重置新值:

```
\setlength{\fboxsep}{1mm}
```

`\fcolorbox 色彩一 色彩二{文本}` 必须先输入宏包 `color` 才能使用本命令, 类似于 `\colorbox`, 文本生成一个带框的 LR 盒子, 以色彩一作为框线颜色, 色彩二作为背景色, 色彩的指定可以都用色彩名, 也可以用同一种模式, 例如:

```
\fcolorbox[rgb]{1,0,0}{0,1,0}{Text}
```

`\fcolorbox{red}{green}{Text}`

`\figurename` 包含浮动图形标题的命令, 在英文环境里, 将打印出 ‘Figure’, 可被重新定义以适应其他语言.

`\fill` 一个弹性长度, 它的自然长度是 0, 但可以无限伸展直至充满整个可用的水平或竖直空间.

`\flat [m]` 产生 b .

`\floatpagefraction` 在一个浮动页中必须被浮动单元充满后才能换新页的部分, 这是一个十进小数, 可用以下命令设置新值:

`\renewcommand{\floatpagefraction}{0.9}`

`\floatsep` 两个同处于顶部或底部的浮动单元间的竖直间隔, 可用 `\setlength` 重置新值:

`\setlength{\floatsep}{12pt plus2pt minus4pt}`

`\flushbottom` 增加段落间的留空使得每一页的最后一行处于同样的位置, 对于文件类 “书籍” (book) 以及选项 `twoside`, 这是标准的设置.

`\fnsymbol{计数器}` 把计数器的当前值用 ‘脚注符号’ 打印:

`*†‡§¶||**††‡‡`

`\fontfamily{族}` 本命令选取字体的族, 对标准 L^AT_EX 而言, 族的可能值为 `cmr`, `cms`, `cmtt`, `cmfi`.

`\fontseries{系列}` 本命令在一个族内选取字体的系列, 对标准 L^AT_EX 而言, 系列的可能值为: `m` (中等粗细), `bx` (粗体).

`\fontshape{形状}` 本命令选取字体的形状, 对标准 L^AT_EX 而言, 形状的可能值为 `n` (正常), `it` (斜体), `sl` (斜体), `sc` (小号大写字母), `u` (直立的意大利体).

`\fontsize{尺寸}{行距}` 本命令选取字体的尺寸, 尺寸是以 `pt` 为单位的整数值, 而行距则给出 `\baselineskip` 的值, 例如:

`\fontsize{12}{14pt}`

`\footnote[数]{文本}` 产生以脚注文本为内容的脚注, 并用可选的数作为编号代替自动编号.

`\footnotemark[数]` 在当前文本产生一个脚注标号, 并用可选的数作为编号代替自动编号, 此命令可用于一些禁用 `\footnote` 的结构里, 如: LR 盒子, 表格, 数学公式等.

`\footnoterule` 这是一个内部命令, 用来在正文与脚注之间画一条水平线, 可被重新定义为:

`\renewcommand{\footnoterule}{\rule{宽度}{高度}\vspace{-高度}}`

`\footnotesep` 在两个脚注间的竖直间距, 可用 `\setlength` 重置新值:

`\setlength{\footnotesep}{6.5pt}`

`\footnotesize` 把字体尺寸转换成 `\footnotesize`, 它比 `\scriptsize` 大, 但比 `\small` 小.

`\footnotetext[数]{脚注文本}` 产生以脚注文本为内容的脚注, 但不在当前正文中添加脚注标号, 页底脚注前面的标号取自计数器 `footnote` 的当前值, 而且标注后不改变此计数器的值, 当给出可选的数时, 则以此数值作为标号, 本命令可与 `\footnotemark` 配合用于一些禁用 `\footnote` 的结构里, 如: LR 盒子, 表格, 数学公式等, 以插入脚注, 不过 `\footnotetext` 命令必须放在这些结构的外面.

`\footskip` 从正文底部到脚注水平线的下缘间的距离, 可用 `\setlength` 重置新值:

`\setlength{\footskip}{25pt}`

`\forall` 产生 \forall .

`\foreignlanguage{语言}{文本}` 在 `babel` 系统里用指定的语言排印一段短的文本.

`\frac{分子}{分母}` `[m]` 生成分式的数学命令.

`\frame{文本}` 围绕文本产生一个没有间隙的框, 如 `\text`, 主要用于 `picture` 环境的 `\put` 或 `\multiput` 命令里.

`\framebox[宽度][位置]{文本}` 产生一个围绕文本的、具有指定宽度的边框, 可选项位置的取值为: `l` 或 `r`, 分别表示文本应在框中居左或居右, 默认值则是居中, 在 L^AT_EX 2_ε 中还可选 `s`, 即把文本伸展到整个宽度.

`\framebox(宽度,高度)[位置]{文本}` `picture` 环境里的绘图元素, 被用在命令 `\put` 或 `\multiput` 的参数中, 它产生一个有指定宽度与高度的方框, 在没有可选参数位置时, 文本放在框的中心, 可选参数位置规定文本在框内的位置: `l`, `r`, `t`, `b` 分别表示左边, 右边, 顶部, 底部, 还可取其中两个字母的组合, 如 `lt` 表示左上角, 在 L^AT_EX 2_ε 中还可选 `s`, 即把文本伸展到整个宽度.

`\frenchspacing` 使用这个命令后, 句点后面不再有额外的留空, 与此相反的命令是 `\nonfrenchspacing`.

`\frontmatter` 在文件类“书籍”(book)中, 本命令用于引入正文前面的内容, 如前言, 目录等, 其作用是关闭命令 `\chapter` 中的章计数功能, 并用罗马数字打印页码.

`\frown` `[m]` 产生 \curvearrowright .

`\fussy` 抵消 `\sloppy` 命令的作用, 使得词间距恢复正常.

`\Gamma` `[m]` 产生 Γ .

`\gamma` `[m]` 产生 γ .

`\gcd [m]` 在数学公式里产生函数名 ‘gcd’, 本命令允许带下标作为下界.

`\ge [m]` 产生 \geq .

`\genfrac{左定界符}{右定界符}{线厚}{字大小}{上部}{下部} [m][a]` 生成一般分式, 线厚是分式线的厚度, 字大小可取值 0-3, 用 `\displaystyle`、`\textstyle`、`\scriptstyle` 和 `\scriptscriptstyle` 表示数学字体的大小, 如为空, 则表示自动设置, 例如 `\binom` 的定义就是:

```
\genfrac{()}{0pt}{}{#1}{#2}
```

`\geq [m]` 产生 \geq .

`\gets [m]` 产生 \leftarrow .

`\gg [m]` 产生 \gg .

`\glossary{条目}` 若在前言中加入了命令 `\makeglossary`, 则本命令写入一条 `\glossaryentry` 命令到 `glo` 文件, 否则什么也不做.

`\glossaryentry{条目}{页码}` 命令 `\glossary` 在 `glo` 文件中写入的内容.

`\graphpaper[数](x,y)(lx,ly)` 宏包 `graphpap` 为 `picture` 环境增加的命令, 它画出一个带有标注的方格坐标系, 其左下角坐标为 (x,y) , 宽度为 lx , 高度为 ly , 每隔指定数个单位画一条线, 第 5 条线是粗线, 数的默认值是 10, 所有参数都必须为整数.

`\grave{x} [m]` 产生数学变量 x 上的重音号: `\grave{a} = \grave{a}`.

`\Grave{x} [m]` 类似于 `\grave`, 不过遇到多重 $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX 重音号时会自动调整位置.

`\guillemotleft` 当 T1 编码被激活时, 产生符号 «.

`\guillemotright` 当 T1 编码被激活时, 产生符号 ».

`\guilsinglleft` 当 T1 编码被激活时, 产生符号 <.

`\guilsinglright` 当 T1 编码被激活时, 产生符号 >.

`\H{x}` 产生重音号: `\H{o} = \ddot{o}`.

`\hat{x} [m]` 产生数学变量 x 上的重音号: `\hat{a} = \hat{a}`.

`\Hat{x} [m]` 类似于 `\hat`, 不过遇到多重 $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX 重音号时会自动调整位置.

`\hbar [m]` 产生 \hbar .

`\headheight` 每一页的页眉高度, 可用 `\setlength` 重置新值:

```
\setlength{\headheight}{25pt}
```

`\headsep` 每一页的页眉底部到正文顶部的竖直距离, 可用 `\setlength` 重置新值:

```
\setlength{\headsep}{0.25in}
```

`\headtoname` 文件类 “信件” (`letter`) 中的命令, 在信件的第二页起的页眉上会出现 ‘To 收信人姓名’, `\headtoname` 的内容就是 To 这个词, 为了适用于其他语

言, 可以对此重新定义.

`\heartsuit [m]` 产生 ♥.

`\height` 等于一个盒子的自然高度 (从基线到顶部的距离) 的长度参数, 它主要用于命令 `\makebox`, `\framebox` 或 `\savebox` 的宽度参数内, 或者用于 `\parbox` 及 `minipage` 环境的高度参数中, 例如:

`\framebox[6\height]{text}`

`\hfill` 水平弹性长度, 其自然长度等于 0, 但可伸展成任意值, 它可用于以空白填满一个行, 实际上这个命令是 `\hspace{\fill}` 的缩写.

`\hline` 本命令在 `array` 与 `tabular` 环境里产生一条贯穿整个表格宽度的水平线.

`\hoffset` 输出页面与打印机设定的打印起始线之间的水平位移, 通常打印起始线与纸边的距离是 `1in`, `\hoffset` 等于 `0pt`, 可以用 `\setlength` 命令重置新值:

`\setlength{\hoffset}{-1in}`

`\hom [m]` 在数学公式里产生函数名 ‘hom’.

`\hookleftarrow [m]` 产生 \hookleftarrow .

`\hookrightarrow [m]` 产生 \hookrightarrow .

`\hrulefill` 用水平直线填满一段空间, 例如: `_____ = \hrulefill`.

`\hspace{宽度}` 产生具有指定宽度的水平空间, 当它位于行首或行尾时则被忽略.

`\hspace*{宽度}` 产生具有指定宽度的水平空间, 即使位于行首或行尾时也不被忽略.

`\huge` 把字体尺寸转换成 `\huge`, 它比 `\Huge` 小, 比 `\LARGE` 大.

`\Huge` 把字体尺寸转换成最大的 `\Huge`, 它比 `\huge` 更大.

`\hyphenation{表} [p]` 为难拆分的词建立的表, 表的形式是:

`hy-phen-a-tion per-mit-ted.`

`\i` 产生 *i*.

`\idotsint [m][a]` 产生 $\int \cdots \int$.

`\iff [m]` 产生 \iff .

`\IfFileExists{文件名}{是}{否}` 测试在 L^AT_EX 输入文件的目录里是否存在此文件名的文件, 若存在, 则执行是提供的命令序列, 否则, 执行否提供的命令序列, 本命令与 `\InputIfFileExists` 类似, 不过这里并不输入.

`\iflanguage{语言}{是}{否}` 在多语言的 `babel` 系统里, 测试语言是否当前选用的语言, 若是, 则执行是提供的命令序列, 否则, 执行否提供的命令序列.

`\ifthenelse{测试}{真}{假}` 本命令要在输入了标准宏包 `ifthen` 后才能使用, 如果逻辑语句测试的执行结果为真, 则执行真提供的命令序列, 否则, 执行假提供的命令序列, 逻辑语句可以是: 关系式 (用 `< = >` 连接的两个数), 奇偶性测试

(`\isodd{整数}`), 两个文本的比较 (`\equal{文本一}{文本二}`), 两个长度的比较 (`\lengthtest{长度一 op 长度二}`, 其中 `op` 是 `< = >` 中之一) 或布尔开关的测试 (`\boolean{开关}`), 这里的布尔开关是由命令 `\newboolean{开关}` 创立的, 并且由命令 `\setboolean{开关}{值}` 置值, 其中的值可以为 `true` 或 `false`, 逻辑语句还可以用逻辑算子 `\and`, `\or` 以及 `\not` 联合, 并用 `\(` 与 `\)` 分组.

测试 `\isodd`, `\lengthtest` 以及 `\boolean` 是 L^AT_EX 2_ε 的新内容.

`\Im [m]` 产生 \Im .

`\imath [m]` 产生 \imath .

`\in [m]` 产生 \in .

`\include{文件名}` 把名为文件名, 扩展名为 `tex` 的文件内容插入当前位置, 注意总是另起一个新页! 本命令与 `\includeonly` 一起, 使得有可能仅处理文件的一部分, 好像其他部分也一起被处理一样.

`\includegraphics[llx, lly][urx, ury]{文件名}` 本命令要输入宏包 `graphics` 才有效, 它输入文件名指定的文件里的图形, 图形边框的坐标是 (llx, lly) (左下角) 以及 (urx, ury) (右上角), 只有这个范围里的图形可接受进一步处理: 缩放或旋转, 文本中也仅为处理后的边框保留空间, 超出这个范围的图形仍被打印出来, 但可能会与相邻文本重叠.

图形边框的坐标可以带有单位, 默认的单位是大点 `bp` ($1/72$ in).

如果边框的坐标被忽略, 则可用别的方法得到信息, 取决于图形文件的类型, 对于 `eps` 文件, 信息可从图形文件本身得到.

如果 `llx` 与 `lly` 未被指定, 则取默认值 0, 也就是说, 如果值给出一组坐标, 则被认为是右上角坐标.

`\includegraphics*[llx, lly][urx, ury]{文件名}` 类似于 `\includegraphics`, 只是边框以外的部分被舍去.

`\includeonly{文件名表} [p]` 只有在文件名表中出现的文件才被 `\include` 命令读入, 其他文件的 `\include` 则被忽略, 表中文件名之间用逗号分隔, 而所有的辅助文件都被输入, 使得页码, 章节号以及交叉参考都保持正确.

`\indent` 下一段落的第一行将被缩进.

`\index{条目}` 若前言中加入了命令 `\makeindex`, 则本命令写入一条 `\indexentry` 命令到 `idx` 文件, 否则什么也不做, 如果这些条目有以下形式:

`\index{主条目}`

`\index{主条目!次条目}`

`\index{主条目!次条目!三级条目}`

则用 `MakeIndex` 文件处理这个 `idx` 文件后可以创立一个 `theindex` 环境, 使得

这些条目按字母次序排列, 并用命令 `\item`, `\subitem` 与 `\subsubitem` 加以组织.

`\indexentry{条目}{页码}` 这是 `\index` 命令写在 `idx` 文件里的条目信息.

`\indexname` 此命令包含索引的标题, 在英文环境里为 'Index', 可根据使用的语言给予重新定义.

`\indexspace` 这是 `theindex` 环境里的命令, 它产生一个空白行.

`\inf [m]` 在数学公式中产生函数名 'inf' 的命令, 可带下标作为下界.

`\infty [m]` 产生 ∞ .

`\intertext{文本} [m][a]` 本命令在多行数学公式内插入居左对齐的文本行, 而不影响其他公式的对齐.

`\input{文件名}` 把名为文件名, 扩展名为 `tex` 的文件内容插入当前位置, 在输入文件内可以含有 `\input` 命令.

`\InputIfFileExists{文件名}{是}{否}` 测试在 L^AT_EX 输入文件的目录里是否存在在此文件名的文件, 若存在, 则执行是提供的命令序列并且输入此文件, 否则, 执行否提供的命令序列.

`\int [m]` 产生 \int .

`\iint [m][a]` 产生 \iint .

`\iiint [m][a]` 产生 \iiint .

`\iiiiint [m][a]` 产生 \iiiiint .

`\intextsep` 表示被插入页面中间的浮动单元与上下文本间的竖直间距, 可用命令 `\setlength` 为其重置新值:

`\setlength{\intextsep}{10pt plus2pt minus3pt}`

`\invisible` 当文件类是 "投影片" (`slides`) 时, 本声明使以下文本用 '隐形墨水' 打印, 也就是说, 它不被打印, 但占据空间, 它的作用范围可持续到所在局部环境的结束, 或遇到 `\visible` 命令, 本命令可用于生成覆盖片.

`\iota [m][a]` 产生 ι .

`\it` 转变为罗马族, 斜体形状, 中等粗细系列 (*Roman, italic, medium*) 的字体属性.

`\itdefault` 本命令定义由 `\itshape` 命令选取的形状属性, 可用 `\renewcommand` 重新定义:

`\renewcommand{\itshape}{it}`

`\item[标签]` 在列表 (`list`) 环境中生成一个标签, 并引出条目的文本, 当没有可选项时, 标签是根据环境的类型生成的, 例如 `enumerate` 环境中是数, 否则, 用选项标签取代.

`\item{条目}` 在 `theindex` 环境里生成一个主条目.

`\itemindent` 在 `list` 环境里, 标签以及每个条目的第一行文本的缩进量, 其标准值为 0pt, 但可用 `\setlength` 为其重置新值:

`\setlength{\itemindent}{1em}`

`\itemsep` 在 `list` 环境里, 不同条目的间距等于 `\parsep` 加上 `\itemsep`, 可用命令 `\setlength` 为其重置新值:

`\setlength{\itemsep}{2pt plus1pt minus1pt}`

`\itshape` 本声明把字体的形状属性改为斜体, 但保留族与系列不变.

`\j` 产生 j .

`\jmath [m]` 产生 j .

`\Join [m]` 产生 \Join .

`\jot` 在 `eqnarray` 或 `eqnarray*` 环境里, 公式行之间的竖直距离, 其标准值等于 3pt, 可用 `\setlength` 为其重置新值:

`\setlength{\jot}{4.5pt}`

`\k{x}` 当 T1 编码被激活时, 产生重音号: `\k{A}=A˘`.

`\kappa [m]` 产生 κ .

`\ker [m]` 在数学公式中产生函数名 ‘ker’ 的命令.

`\kill` 在 `tabbing` 环境里, 放在样本行的末尾, 使得样本行只用于设定指标位的位置, 而不被打印出来.

`\L` 产生 L .

`\l` 产生 l .

`\label{标志}` 在当前位置设立一个具有指定名称的标志, 以供文件中其他地方 (可以在它的前面) 用命令 `\ref{标志}` 引用, 打印出章节号、方程号或图号等, 也可用命令 `\pageref{标志}` 引用, 打印页号.

`\labelnumn` 用在 `enumerate` 环境里, 这是一组命令, 可生成嵌套层次的标准标签, n 的取值为 i, ii, iii, iv , 例如: 命令

`\renewcommand{\labelnumii}{\arabic{enumii}.)}`

把 `enumerate` 环境的第二级标准标签修改成 1.), 2.), 等等.

`\labelitemn` 用在 `itemize` 环境里, 这是一组命令, 可生成嵌套层次的标准标签, n 的取值为 i, ii, iii, iv , 例如: 命令

`\renewcommand{\labelitemi}{\text{\rightarrow}}`

把 `itemize` 环境的最外层标准标签修改成 \Rightarrow .

`\labelsep` 在 `list` 环境里, 标签与文本间的距离, 可用 `\setlength` 为其重置新值:

`\setlength{\labelsep}{5pt}`

`\labelwidth` 在 `list` 环境里, 放标签的盒子的宽度, 可用 `\setlength` 为其重置新

值:

`\setlength{\labelwidth}{2.2cm}`

`\Lambda` [m] 产生 Λ .

`\lambda` [m] 产生 λ .

`\langle` [m] 产生 \langle .

`\language{数}` 在 T_EX 3.0 版以后, 不同语言的拆分词模式用不同数字作标志, 在执行 `initex` 时, 必须先给出命令 `\language{数}`, 让程序选取适当的分词模式输入格式文件.

`\large` 把字体尺寸转换成 `\large`, 它比 `\Large` 小, 比 `\normalsize` 大.

`\Large` 把字体尺寸转换成 `\Large`, 它比 `\LARGE` 小, 比 `\large` 大.

`\LARGE` 把字体尺寸转换成 `\LARGE`, 它比 `\huge` 小, 比 `\Large` 大.

`\LaTeX` 产生 L^AT_EX.

`\LaTeXe` 产生 L^AT_EX 2_ε.

`\lceil` [m] 产生 \lceil .

`\ldots` [m] 产生

`\le` [m] 产生 \leq .

`\leadsto` [m] 产生 \leadsto .

`\left` 定界符 [m] 调整定界符的大小使其适应夹在 `\left ... \right` 内的公式的高度, 例如: `\left[`, 如果另一端没有明显的定界符与之配对, 则可使用空定界符 (`\right.`).

`\Leftarrow` [m] 产生 \Leftarrow .

`\leftarrow` [m] 产生 \leftarrow .

`\leftharpoondown` [m] 产生 \leftharpoondown .

`\leftharpoonup` [m] 产生 \leftharpoonup .

`\lefteqn{公式}` [m] 环境 `eqnarray` 的内部命令, 此命令虽然打印出公式, 但又把它的宽度当成 0, 使它不影响列的宽度, 这个命令主要用于多行公式的第一行.

`\leftmargin` 在 `list` 环境里, 这是环境内文本的左边界与外部正文的左边界相比的缩进值, 可用命令 `\setlength` 为其重置新值. 此外, 在多层次的 `list` 环境里, 可以在此命令的后面添加 `i...iv` 表示各层次的缩进值, 例如:

`\setlength{\leftmarginiii}{0.5cm}`

`\Leftrightarrow` [m] 产生 \Leftrightarrow .

`\leftrightharpoonup` [m] 产生 \leftrightharpoonup .

`\leftroot{数}` [m][a] 使 `\sqrt` 的根指数稍微左移一点, 例如:

`\sqrt[\leftroot{-1}]{\uproot{3}{\beta}}{k}`

- `\leq [m]` 产生 \leq .
- `\lfloor [m]` 产生 \lfloor .
- `\lg [m]` 在数学公式中产生函数名 ‘lg’ 的命令.
- `\lhd [m]` 产生 \lhd .
- `\lim [m]` 在数学公式中产生函数名 ‘lim’ 的命令, 可带下标作为下界.
- `\liminf [m]` 在数学公式中产生函数名 ‘liminf’ 的命令, 可带下标作为下界.
- `\limits [m]` 把上下界放在相应符号的上方与下方, 而正常情况是放在后方的.
- `\limsup [m]` 在数学公式中产生函数名 ‘limsup’ 的命令, 可带下标作为下界.
- `\line($\Delta x, \Delta y$){长度}` 本命令是 `picture` 环境里的一个图形元素, 通常作为 `\put` 或 `\multiput` 的参数, 其作用是绘制水平或竖直直线段以及某些特定斜率的斜线段. 对于水平或竖直直线段, 长度就是线段的实际长度 (以 `\unitlength` 为单位), 对于斜线段, 长度是它在 x 轴上的投影 (即水平位移). ($\Delta x, \Delta y$) 给出了线段的斜率, 其中 Δx 与 Δy 是一对互素整数, 满足 $-6 \leq \Delta x \leq 6, -6 \leq \Delta y \leq 6$.
- `\linebreak[n]` 鼓励在此处换行, 并使此行在左右两端都对齐, 换行的迫切性由整数 n 的值确定, $0 \leq n \leq 4$, 4 表示强迫换行.
- `\linethickness{厚度}` 在 `picture` 环境里设置水平或竖直直线段的厚度, 厚度是带有单位的长度, 如: 1.2mm.
- `\listfigurename` 此命令含有插图目录的标题, 在英语中定义为 ‘List of Figures’, 但可被重新定义以适用于其他语言.
- `\listfiles [p]` 如果在前言里给出了此命令, 则在处理过程中所有读入的文件都会在程序结束后显示在监视器上, 同时也被写入纪录文件. 文件列表中包括版本号, 日期以及其他附加信息.
- `\listoffigures` 产生一个插图目录, 其条目内容取自 `figure` 环境中的 `\caption` 命令.
- `\listoftables` 产生一个表格目录, 其条目内容取自 `table` 环境中的 `\caption` 命令.
- `\listparindent` 在 `list` 环境里, 段落的第一行的缩进量, 可用 `\setlength` 命令重置新值:
- `\setlength{\listparindent}{1em}`
- `\listtablename` 此命令含有表格目录的标题, 在英语中定义为 ‘List of Tables’, 但可被重新定义以适用于其他语言.
- `\ll [m]` 产生 \ll .
- `\ln [m]` 在数学公式中产生函数名 ‘ln’ 的命令.

`\LoadClass`[选项]{类}[版本号] [p] 本命令只能出现在类文件里, 以输入另一个类文件, 在一个类文件里只能使用一次, 被输入文件的扩展名必须是 `cls`, 而且在 `\documentclass` 命令中出现的选项不作为整体选项传递.

选项版本号的格式是: `yyyy/mm/dd`, 例如: `1994/08/01`. 如果类文件的日期比它早, 就会打印一个警告信息.

`\location`{数} 当文件类是“信件”(letter)时, 输入发信人的房间号码. 在标准 L^AT_EX letter 类中, 仅当 `\address` 没有出现时才输出这个数, 其目的是用于公司的信头.

`\log` [m] 在数学公式中产生函数名 ‘log’ 的命令.

`\Longleftarrow` [m] 产生 \longleftarrow .

`\longleftarrow` [m] 产生 \longleftarrow .

`\Longleftrightarrow` [m] 产生 \longleftrightarrow .

`\longleftrightarrow` [m] 产生 \longleftrightarrow .

`\longmapsto` [m] 产生 \longmapsto .

`\Longrightarrow` [m] 产生 \longrightarrow .

`\longrightarrow` [m] 产生 \longrightarrow .

`\lq` 产生 ‘, 与 ‘ 符号完全相同.

`\lvert` [m][a] 产生 | (左定界符).

`\lVert` [m][a] 产生 || (左定界符).

`\mainmatter` 在文件类“书籍”(book)中, 本命令宣告进入书籍的正文, 它把页码计数器重置为 1, 且用阿拉伯数字打印, 恢复 `\chapter` 命令中的章节计数功能, 总之, 消除 `\frontmatter` 命令的后果.

`\makebox`[宽度][位置]{文本} 产生一个有指定宽度且包含文本的盒子, 选项位置决定了文本在盒子中的位置, `l`或`r`分别表示居左或居右, 默认是居中, L^AT_EX 2_ε中增加了一个 `s`, 表示伸展到整个宽度.

`\makebox`(宽度, 高度)[位置]{文本} 这是 `picture` 环境里的一个绘图命令, 用作 `\put` 或 `\multiput` 命令的参数, 它产生一个有指定宽度与高度且包含文本的盒子. 选项位置是指文本在盒子内部的位置, 它们可以是居左(`l`), 居右(`r`), 居顶(`t`)或居底(`b`), 也可以是两者的组合, 如 `lt`, 默认值是居中. L^AT_EX 2_ε中增加了一个 `s`, 表示伸展到整个宽度.

`\makeglossary` [p] 激活正文中的 `\glossary` 命令.

`\makeindex` [p] 激活正文中的 `\index` 命令.

`\makelabel`{文本} 被命令 `\item` 调用的内部命令, 它在 `list` 环境里产生一个内容为文本的标签.

`\makelabels` 当文件类是“信件”(letter)时, 利用`\begin{letter}`环境里的条目产生地址标签.

`\MakeShortVerb{\字母}` 当输入了标准宏包`shortvrb`后, 本命令使字母成为`\verb`字母的缩写, 于是夹在两个字母中间的文本都被按照字符的本来意义打印. 例如在发出命令`\MakeShortVerb{\|}`后, 输入`| \cmd{|}`等价于输入`\verb|\cmd{|}`, 其打印输出为`\cmd{|}`. 在发出命令`\DeleteShortVerb{\|}`后, 字符`|`又恢复其本来意义.

`\maketitle` 利用`\author`, `\title`以及`\date`, `\thanks`中包含的信息产生一个标题页.

`\mapsto [m]` 产生 \mapsto .

`\marginpar[左文本]{右文本}` 在页面的右边空白处产生内容为右文本的注记. 在双页格式中遇到偶数页时, 页边注记打印在左方, 这时就用可选项左文本替代. 在双栏格式里, 页边注记总是打印在外侧, 这时也用左文本代替左页边的注记.

`\maginparpush` 两个页边注记间的最小竖直距离, 可用`\setlength`重置新值.

`\maginparsep` 正文与页边注记间距离, 可用`\setlength`重置新值.

`\maginparwidth` 页边注记所在盒子的宽度, 可用`\setlength`重置新值.

`\markboth{左边}{右边}` 在双页格式里, 如果把页面格式选为`myheadings`, 或当页面格式`headings`的自动页眉设置被改变后, 本命令设置左右页眉的文本.

`\markright{左边}{右边}` 如果把页面格式选为`myheadings`, 或当格式`headings`的自动页眉设置被改变后, 本命令设置页眉的文本. 如果是双页格式, 则只设置右页眉.

`\mathbf{文本} [m]` 在数学模式里用粗体(`\bfseries`)打印文本, 其中的空格被忽略.

`\mathcal{文本} [m]` 在数学模式里用书写体打印文本, 本命令取代了L^AT_EX2.09中的`\cal`.

`\mathindent` 当选项`fleqn`被选中后, 行间公式在左边的缩进量, 可用`\setlength`为其重置新值:

`\setlength{\mathindent}{25pt}`

`\mathit{文本} [m]` 在数学模式里用斜体(`\itshape`)打印文本, 它与`\mathnormal`的区别可从下例看出: 比较`mathit`与`mathnormal`.

`\mathnormal{文本} [m]` 在数学模式里用数学字体打印文本, 它取代了L^AT_EX2.09中的`\mit`.

`\mathring{x} [m]` 产生数学模式里的重音号: `\mathring{a} = \mathring{a}`.

`\mathrm{文本}` [m] 在数学模式里用罗马体 (`\rmfamily`) 打印文本, 其中的空格被忽略.

`\mathsf{文本}` [m] 在数学模式里用无衬线字体 (`\sffamily`) 打印文本, 其中的空格被忽略.

`\mathtt{文本}` [m] 在数学模式里用打字机字体 (`\ttfamily`) 打印文本, 其中的空格被忽略.

`\mathversion{变体}` 选取当前的数学变体, 可能的值为 `bold` 与 `normal`, 用命令 `\DeclareMathVersion` 可创立新的变体.

`\max` [m] 在数学公式中产生函数名 ‘max’ 的命令, 可带下标作为下界.

`\mbox{文本}` 围绕文本建立一个 LR 盒子.

`\mddefault` 定义了由命令 `\mdseries` 选取的字体系列, 可用 `\renewcommand` 予以重定义:

```
\renewcommand{\mddefault}{m}
```

`\mdseries` 本声明不改变当前字体的族与形状, 但转变成中等粗细 `medium` 序列.

`\medskip` 插入值为 `\medskipamount` 的竖直间隔, 参见 `\bigskip`, `\smallskip`.

`\medskipamount` 用于 `\medskip` 的竖直间隔的标准值, 可用命令 `\setlength` 加以改变:

```
\setlength{\medskipamount}{3ex plus1ex minus1ex}
```

`\medspace` [m][a] 这是 `\:` 的别名, 用在数学公式中插入中等大小的空白.

`\MessageBreak` 在出错信息、警告信息或版本注解信息的文本中产生强迫换行.

`\mho` [m] 产生 \mathcal{U} .

`\mid` [m] 产生 $|$.

`\min` [m] 在数学公式中产生函数名 ‘min’ 的命令, 可带下标作为下界.

`\mit` 这是 L^AT_EX 2.09 的命令, 表示选取数学斜体, 在 L^AT_EX 2_ε 中已被 `\mathnormal` 取代.

`\mod{变量}` [m][a] 在输入宏包 `amsopn` 与 `amsmath` 后, 本命令数学公式中产生函数名 ‘mod’, 例如:

```
y\mod{a+b}=y \mod a + b
```

`\models` [m] 产生 \models .

`\mp` [m] 产生 \mp .

`\mspace{长度}` [m][a] 在数学公式里插入指定长度的空白, 长度单位是 `mu` ($= 1/18\text{em}$), 例如: `\mspace{-4mu}`

`\mu` [m] 产生 μ .

`\multicolumn{列数}{列格式}{文本}` 在 `tabular` 和 `array` 环境里把指定个数的列组合成单个列, 列格式指定组合后单个列的格式, 可取值 `l`, `c`, `r` (只能一个) 以及添加竖线 `|`.

`\multiplot(x,y)(\Delta x,\Delta y){个数}{图形元素}` 在 `picture` 环境里以 (x,y) 作为图形元素的基准点坐标, 以 $(\Delta x,\Delta y)$ 作为平移向量, 画出指定个数的图形元素.

`\multinegap [m][a]` 在 $\mathcal{A}\mathcal{M}\mathcal{S}$ -L^AT_EX 的 `multline` 环境里产生的多行公式的左右缩进量, 初始值是 10pt, 用户可重置新值.

`\nabla [m]` 产生 ∇ .

`\name{发信人}` 在文件类 “信件” (`letter`) 中用以输入发信人姓名.

`\natural [m]` 产生 \natural .

`\nearrow [m]` 产生 \nearrow .

`\NeedsTeXFormat{格式}[版本] [p]` 声明为了处理这个文件所需的 T_EX 格式, 这应该是文件的第一个语句. 到目前为止, 格式的合法值仅有 `LaTeX2e`. 可选项版本应该输入一个格式为 `yyyy/mm/dd` 的日期, 指明与本文件兼容的最早的版本日期, 例如:

```
\NeedsTeXFormat{LaTeX2e}[1994/06/01]
```

`\neg [m]` 产生 \neg .

`\negmedspace [m][a]` 在数学公式中插入一个负的中等大小的空白.

`\negthickspace [m][a]` 在数学公式中插入一个负的较大的空白.

`\negthinspace [m][a]` 这是 `\!` 的别名, 在数学公式中插入一个负的小间隙.

`\newboolean{开关}` 使用本命令必须先输入标准 L^AT_EX 宏包 `ifthen`, 它创建一个新的布尔开关, 开关的值由命令 `\setboolean{开关}{值}` 指定, 这里的值可取 `true` 或 `false`. 检测开关的值的命令是 `\boolean{开关}`, 这个检测命令可用于 `\ifthenelse` 或 `\whiledo` 的测试部分.

`\neq [m]` 产生 \neq .

`\newcommand{\命令名}[参量个数][选项]{命令内容}` 定义用户命令 `\命令名` 使其具有指定的命令内容, 选项参量个数 ≤ 9 , 命令内容中用 `\#1, \dots, \#n` 参量个数代表这些参量. 如果本命令还带有第二个选项 (L^AT_EX 2.09 无此功能), 则说明新命令的第一个参量是可选的, 而且当它不被给出时以这里的选项作为默认值.

`\newcommand*{\命令名}[参量个数][选项]{命令内容}` 类似于 `\newcommand`, 不过命令内容必须是短的, 即在命令内容中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\newcounter{计数器名}[主计数器名]` 建立一个具有指定计数器名的新计数器.

可选参数主计数器名是一个已有的计数器,使得每当主计数器的值变化时,新建计数器被置零,也就是说,使新计数器成为从计数器.

`\newenvironment{环境名}[参量个数][选项]{进入命令}{退出命令}` 定义有指定环境名的用户环境,当遇到`\begin`命令进入此环境时,就执行进入命令,遇到`\end`命令退出时,则执行退出命令. 选项参量个数 ≤ 9 ,在进入命令中用`#1,...`参量个数代表这些参量. 如果本命令还带有第二个选项(L^AT_EX 2.09 无此功能),则说明`\begin`命令的第一个参量是可选的,而且当它不被给出时以这里的选项作为默认值.

`\newenvironment*{环境名}[参量个数][选项]{进入命令}{退出命令}` 类似于`\newenvironment`,不过进入命令必须是短的,不能出现新的段落,也就是说,不能有`\par`及空白行.

`\newfont{\字体命令}{字体文件 scaled 放大倍数}` 在`\字体命令`与外部的(经过适当放缩的)字体文件之间建立一个关联,执行这个`\字体命令`后,基线间隔`\baselineskip`以及行距等都保持原样.

`\newlength{\长度命令}` 创建一个名为`\长度命令`的新命令,其初始值为0pt,可用`\setlength`为其重置新值,例如:

`\setlength{\lengthcmd}{1mm}`

其中的长度必须带有单位,也可以是弹性长度.

`\newline` 结束当前行,但不向右对齐.

`\newpage` 结束当前页,剩下部分保持空白.

`\newsavebox{\盒子名}` 创建一个名为`\盒子名`的新的存储盒子,可用`\savebox`命令把LR盒子保存在其中.

`\newtheorem{定理环境名}[编号]{标题}`

`\newtheorem{定理环境名}{标题}[主计数器名]` 定义一个名为定理环境名的新的定理环境,每当这个环境被调用时,先用黑体字打印标题(例如 **Theorem**, **Proposition** 等),再打印一个自动生成的序号,然后用斜体打印环境内部的文本. 选项编号是另一个定理环境的名字,它与当前的环境共用同一个序号计数器. 另一个可选项主计数器名是章节计数器的名称,例如 `chapter` 或 `section`,每当主计数器值改变时,就重置定理计数器的值,使得定理计数器成为从计数器. 两个可选项每次只能选一个.

`\NG` 当 T1 编码被激活时,产生字母 Ω .

`\ng` 当 T1 编码被激活时,产生字母 η .

`\ni [m]` 产生 \ni .

`\nocite{引用词}` 参考文献数据库里具有引用词的条目将被包含在文献目录里,

即使在文本中未被引用. 如果用 `\nocite{*}`, 则数据库里的所有条目都将被包含在文献目录里.

`\nocorr` 与字体命令 `\emph`, `\textsl`, `\textit` 合用, 以取消自动的斜体校正, 如:

`\nocorr\emph{emphatic text}` 或

`\emph{emphatic text\nocorr}`

`\nofiles [p]` 不输出辅助文件 `aux`, `glo`, `idx`, `lof`, `lot` 以及 `toc`.

`\noindent` 使下一段落的第一行不缩进.

`\nolimits [m]` 把上下界放在符号的后面, 而不是放在上下.

`\nolinebreak[n]` 建议此处不要换行, 其迫切程度由整数 n 确定, $0 \leq n \leq 4$, n 取 4 时相当于没有选项, 意为禁止换行.

`\nonfrenchspacing` 与 `\frenchspacing` 相反, 使用这个命令后, 句子后面加上额外的留空.

`\nonumber [m]` 在 `eqnarray` 环境的多行公式里, 有此命令的行将没有方程编号.

`\nopagebreak[n]` 建议不要在此处换页, n 取 0 到 4 的整数, 迫切性随 n 的增加而增大. $n = 4$ 时相当于不含可选项的命令, 禁止换页.

`\normalcolor` 如果输入了宏包 `color`, 本命令把文字的颜色重置为前言末尾的颜色, 通常是黑色. 前言中的 `\color` 命令可以改变‘标准’色彩. 本命令主要用于 L^AT_EX 内部的宏包中, 使得在打印标题后可以重置文本的颜色, 其他与 `color` 相容的宏包也能使用它.

`\normalfont` 本声明把字体转换为默认的族、形状和系列.

`\normalmarginpar` 取消命令 `\reversemarginpar` 的作用, 使页边注记放在标准位置, 即右边或外侧.

`\normalsize` 本命令把字体尺寸转换成 `\normalsize`, 也就是 `\documentclass` 或 `\documentstyle` 命令的选项中选取的大小, 它比 `\large` 小, 比 `\small` 大.

`\not [m]` 使后面的比较符号取否定的意义: `\not\cong = \ncong`.

`\notag{标记} [m][a]` 用在 \mathcal{AMS} -L^AT_EX 的对齐环境中, 取消公式自动编号.

`\notesname` 含有注记命令 `\notes` 的标题的命令, 在英语中定义为‘notes’, 但可被重新定义以适用于其他语言.

`\notin [m]` 产生 \notin .

`\nu [m]` 产生 ν .

`\numberwithin{计数器}{主计数器} [a]` 把计数器重新定义为另一主计数器的子计数器, 也就是说, 每当主计数器的值增加时, 此计数器就置零. 打印计数器的值时一起打印主计数器的值, 这条命令通常用来使论文中的方程能与节一起计数, 如:

`\numberwithin{equation}{section}`

`\O` 产生 \emptyset .

`\o` 产生 \emptyset .

`\oddsidemargin` 在文件类“书籍”(book)中,或其他文件类但选项 `twoside` 被选取时,本命令设置奇数页的左页边宽,而在其余情形,本命令设置所有页的左页边宽.可用 `\setlength` 命令重置新值,例如:

`\setlength{\oddsidemargin}{1.5cm}`

`\odot` [m] 产生 \odot .

`\OE` 产生 \mathbb{E} .

`\oe` 产生 \mathfrak{e} .

`\oint` [m] 产生 \oint .

`\Omega` [m] 产生 Ω .

`\omega` [m] 产生 ω .

`\ominus` [m] 产生 \ominus .

`\onecolumn` 开始一个新页,并从双栏格式转换成单栏格式.

`\onlynotes{数表}` 在文件类“投影片”(slides)中,本命令出现在前言中,使得只产生数表中出现的数字对应的注记,本命令的作用与 `\onlyslides` 类似.

`\onlyslides{数表}` 在文件类“投影片”(slides)中,本命令出现在前言中,使得只产生数表中出现的数字对应的投影片.数表中的数以逗号分隔,且可含有以连字符表示的范围,如:

`\onlyslides{4,10-13,23}`

`\opening{称呼语}` 在“信件”(letter)类的 letter 环境里,它设置了信件开头的称呼语,例如: `\opening{Dear George,}`.

`\oplus` [m] 产生 \oplus .

`\OptionNotUsed` [p] 本命令只能用于选项的定义中,特别是默认选项的定义中,它声明 `\CurrentOption` 是未经处理的. 本命令用于这样的情况: 由于某些原因,例如缺少必要的文件,默认的选项可能会失败,这时本命令就通知 L^AT_EX, 所需的选项尚有待处理.

`\oslash` [m] 产生 \oslash .

`\otimes` [m] 产生 \otimes .

`\oval(宽度,高度)[部分]` 本命令是 `picture` 环境里的图形元素,产生一个具有指定宽度与高度的圆角矩形,可选参数部分可取值 `t`, `b`, `l`, `r`, 分别表示只画上半部,下半部,左半部或右半部,如果取其中两个的组合,例如 `tl` 或 `lt`,则表示画左上角的四分之一部分. 本命令一般用作 `\put` 或 `\multiput` 命令的参数.

`\overbrace{公式} [m]` 在公式的上面产生一个水平的花括号, 以上标形式出现在本命令后面的数学符号将被居中放在花括号的上方. 例如:

$$\overbrace{a+b} = a + b$$

$$\overbrace{x+y+z}^{\xi\eta\zeta} = x + y + z$$

`\overleftarrow{公式} [m][a]` 在公式的上面画一条指向左方的长箭头.

`\overleftrightharpoon{公式} [m][a]` 在公式的上面画一条左右两边都有箭头的水平线.

`\overline{公式} [m]` 在公式的上面产生一条水平直线:

$$\overline{a-b} = a - b$$

`\overrightarrow{公式} [m][a]` 在公式的上面画一条指向右方的长箭头.

`\overset{字符}{\符号} [m][a]` 把字符按上标的字体放在 `\符号` 上方:

$$\overset{\alpha}{\longrightarrow} = \overset{\alpha}{\longrightarrow}$$

`\P` 产生 ¶.

`\PackageError{宏包名}{出错信息}{帮助} [p]` 本命令只能出现在宏包文件中, 它向监视器以及纪录文件发出以宏包名为标号的出错信息, 中断进程并等待用户的反应, 如果用户键入 H<回车>, 就把帮助打印出来, 在出错信息和帮助中都可用 `\MessageBreak` 命令换行, 用 `\space` 强制产生空格, 并可用 `\protect` 冠在一个命令前面, 使得这个命令不被解释执行而是把它的名字打印出来.

`\PackageInfo{宏包名}{信息} [p]` 类似于 `\PackageWarningNoLine`, 不过信息不在监视器上显示, 只是写入纪录文件.

`\PackageWarning{宏包名}{警告信息} [p]` 本命令只能出现在宏包文件中, 它向监视器以及纪录文件发出以宏包名为标号的警告信息, 同时给出输入文件的当前行号, 并继续进行下去, 警告信息可按 `\PackageError` 同样的方法格式化.

`\PackageWarningNoLine{宏包名}{警告信息} [p]` 类似于 `\PackageWarning`, 只是不打印当前行号.

`\pagebreak[n]` 鼓励在此处换页, 换页的迫切性由整数 n 的值确定, $0 \leq n \leq 4$, 4 表示强迫换页, 相当于不含选项的命令.

`\pagecolor` 色彩 必须先输入宏包 `color` 才能使用本命令, 它设定从本页开始的背景色, 以后的页都使用这种背景色, 直至再遇到命令 `\pagecolor` 为止. 色彩的定义可参看 `\color`.

`\pagename` 本命令用于文件类“信件”(letter)中, 用在从第二页起冠在页码前面. 在英文环境中是‘Page’, 可根据使用的语言加以改变.

`\pagenumbering{格式}` 确定页码的格式, 并把页计数器置 1. 格式的可能值是 `arabic`, `roman`, `Roman`, `alph`, `Alph`.

- `\pageref{标记}` 打印标记所在处的页码, 标记由命令 `\label{标记}` 定义.
- `\pagestyle{格式}` [p] 确定页格式, 也就是每页的页眉与页脚的内容, 格式的可能值为: `plain`, `empty`, `headings`, `myheadings`.
- `\paperheight` 由 `\documentclass` 命令中的页面大小选项所确定的页面的总高度, 在默认的 `lettersize` 选项下, 这是 11in; 若选 `a4paper`, 则为 29.7cm. 选项 `landscape` 交换 `\paperwidth` 与 `\paperheight` 的值.
- `\paperwidth` 由 `\documentclass` 命令中的页面大小选项所确定的页面宽度, 在默认选项 `lettersize` 下, 这是 8.5in; 若选 `a4paper`, 则为 21cm. 选项 `landscape` 交换 `\paperwidth` 与 `\paperheight` 的值.
- `\par` 结束当前段落, 开始一个新的段落. 本命令相当于一个空白行.
- `\paragraph[短标题]{标题}` 章节层次中的倒数第二级, 介于 `\subsubsection` 与 `\subparagraph` 之间, 它在当前小节编号数的后面添加自动生成的段落序号, 后面再打印标题. 可选项短标题用在目录中代替标题.
- `\paragraph*{标题}` 类似于 `\paragraph`, 只是没有编号, 也不在目录中出现.
- `\parallel` [m] 产生 \parallel .
- `\parbox[位置][高度][内部位置]{宽度}{文本}` 产生具有指定宽度的竖直盒子, 其中的文本保持左右对齐, 这个盒子相对于周围环境的竖直位置由可选项位置确定: `t` 表示与顶上的行对齐, `b` 表示与底下的行对齐, 默认是与中间对齐. 另两个可选项 (L^AT_EX 2.09 无此功能): 高度给出盒子的总高度, 内部位置给出文本在盒子内部的竖直位置, 可能的值为: `t` 表示居顶, `b` 表示居底, `c` 表示居中, `s` 伸展到整个高度, 其默认值取外部位置指定的值. 高度中可包含下列参数: `\height`, `\depth`, `\width` 与 `\totalheight`.
- `\parindent` 段落第一行的缩进量, 可用 `\setlength` 为其指定新值:
`\setlength{\parindent}{1.5em}`
- `\parsep` 在 `list` 环境里段落间的竖直间隔, 可用 `\setlength` 为其指定新值:
`\setlength{\parsep}{2pt plus1pt minus1pt}`
- `\parskip` 段落间的竖直距离, 可用 `\setlength` 为其重置新值:
`\setlength{\parskip}{3pt plus1pt minus2pt}`
- `\part[短标题]{标题}` 章节层次中的最高级, 它开始新的‘篇’(Part), 自动生成一个编号, 后接标题, 后面的节号不受篇号的影响. 如果给出了选项短标题, 则它在目录里将取代标题.
- `\part*{标题}` 类似于 `\part`, 不过它没有编号, 也不出现在目录中.
- `\partial` [m] 产生 ∂ .

`\partname` 此命令包含篇的标题, 在英文环境里是 ‘Part’, 可根据不同语言加以修改.

`\partopsep` 在 list 环境里, 如果有一个空白行出现在此环境的前面或后面, 本命令给出了附加的竖直空间. 可用 `\setlength` 命令重置新值:

`\setlength{\partopsep}{2pt plus1pt minus1pt}`

`\PassOptionsToClass{选项}{类}` [p] 把选项传送到后面要用 `\LoadClass` 命令输入类文件中, 本命令必须出现在类文件或被类文件读入的文件中, 它可被用在选项的定义中, 或用在配置文件中, 以激活某个选项.

`\PassOptionsToPackage{选项}{宏包}` [p] 把选项传送到后面用 `\RequirePackage` 命令输入的宏包文件中, 本命令可出现在类文件或宏包文件中, 它被用在选项的定义中, 或用在配置文件中, 以激活某个选项.

`\perp` [m] 产生 \perp .

`\Phi` [m] 产生 Φ .

`\phi` [m] 产生 ϕ .

`\Pi` [m] 产生 Π .

`\pi` [m] 产生 π .

`\pm` [m] 产生 \pm .

`\pmb{符号}` [m][a] 在输入了宏包 `amsmath` 或 `amsbsy` 后, 本命令用多次重复打印的方法把符号打印成黑体.

`\pmod{变量}` [m] 在数学公式里产生带括号的函数名 ‘mod’:

$y \pmod{a+b} = y \pmod{a+b}$

`\pod{变量}` [m][a] 在输入了宏包 `amsmath` 或 `amsopn` 后, 本命令类似于 `\pmod`, 但不出现函数名 ‘mod’:

$y \pod{a+b} = y (a+b)$

`\poptabs` 在 `tabbing` 环境里, 恢复前面用 `\pushtabs` 命令存储起来的制表位.

`\pounds` 产生 \pounds .

`\Pr` [m] 在数学公式里产生函数名 ‘Pr’, 而且可带下标作为下界.

`\prec` [m] 产生 \prec .

`\preceq` [m] 产生 \preceq .

`\prefacename` 本命令包含由命令 `\preface` 产生的标题, 在英文环境里是 ‘Preface’, 可根据不同的语言予以重定义.

`\prime` [m] 产生 \prime (同字符 ‘’).

`\printindex` 文件 `makeidx.sty` 里定义的命令, 在用程序 `Makeindex` 处理了 `idx` 文件后, 本命令可生成 `theindex` 环境.

`\ProcessOptions [p]` 在类文件或宏包文件里, 本命令通过执行 `\ds@` 命令来处理被选中的选项, 按照这些命令被定义的次序执行, 然后这些 `\ds@` 被删除.

`\ProcessOptions*` [p] 类似于 `\ProcessOptions`, 不过它是按照选项被选中的次序执行 `\ds@` 命令, 这相当于 L^AT_EX 2.09 的命令 `\@options`, 为了保持兼容性, 这条命令仍然有效.

`\prod [m]` 产生 \prod .

`\propto [m]` 产生 \propto .

`\protect` 当一条脆弱命令被冠以 `\protect` 后, 就能被用于移动命令中, 例如:

```
\section{The \protect\pounds{ } Sign}
```

`\providecommand{\命令名}[参量个数][选项]{命令内容}` 类似于 `\newcommand`, 不过当 `\命令名` 已经存在时, 命令内容被忽略.

`\providecommand*{\命令名}[参量个数][选项]{命令内容}` 类似于不带星号的命令 `\providecommand`, 不过命令内容必须是短的, 即在命令内容中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\ProvidesClass{类}[版本] [p]` 本命令出现在类文件的头部, 声明类名及其版本, 供输入类文件的命令 `\documentclass` 或 `\LoadClass` 核对. 选项版本分成三个部分: 日期, 版本号以及其他信息, 例如:

```
\ProvidesClass{thesis}[1995/01/25 v2.8 U of City]
```

`\ProvidesFile{文件名}[版本]` 本命令出现在一般文件的头部, 声明其名字及版本. 当用命令 `\input` 输入此文件时, 不进行任何检查, 但当 `\listfiles` 被激活时, 这些信息会被显示出来. 这个命令与其他 `\Provides...` 命令不同, 不必限制在前言内.

`\ProvidesPackage{类}[版本] [p]` 本命令出现在宏包文件的头部, 声明宏包名及其版本, 供输入宏包文件的命令 `\usepackage` 或 `\RequirePackage` 核对. 选项版本分成三个部分: 日期, 版本号以及其他信息, 例如:

```
\ProvidesPackage{notes}[1995/02/13 1.2 G. Smith]
```

`\ProvideTextCommand{\命令}{编码}[参数个数][选项]{定义} [p]` 类似于命令 `\providecommand`, 定义 `\命令`, 不过只有当编码是当前字体的编码时, 此定义才有效, 而且如果此 `\命令` 对于编码已有定义, 就不再重新定义它.

`\ProvideTextCommandDefault{\命令}{编码}[参数个数][选项]{定义} [p]` 类似于 `\DeclareTextCommandDefault`, 不过若 `\命令` 对于默认编码已有定义, 就不再重新定义它.

`\ps` 文本 用于“信件”(letter)文件类中, 在信末加入一个附言.

`\Psi [m]` 产生 Ψ .

`\psi [m]` 产生 ψ .

`\pushtabs` 在 `tabbing` 环境里, 存储当前的制表位, 它可被 `\poptabs` 命令恢复.

`\put(x,y){图形元素}` 在 `picture` 环境里把图形元素放在以 (x,y) 为参考点的位置上.

`\qbezier[点数](x1,y1)(x2,y2)(x3,y3)` 本命令应被用于 `picture` 环境内, 生成一条端点为 (x_1,y_1) , (x_3,y_3) 的二次 Bézier 曲线, 以 (x_2,y_2) 作为控制点. 当选项点数给出时, 此曲线由 (点数 + 1) 个点构成, 否则由程序自动确定所需的点数. 除了点数作为选项外, 其余均同 `\bezier`.

`\quad` 插入长为 1em 的水平空白.

`\qqquad` 插入长为 2em 的水平空白.

`\quotedblbase` 当 T1 编码被激活时, 产生符号 „.

`\quotesinglbase` 当 T1 编码被激活时, 产生符号 ,.

`\r{x}` 产生一个重音号: $\text{\r{o}} = \text{\r{ö}}$.

`\raggedbottom` 如果没有选取选项 `twoside`, 则这是“文章”(article)、“报告”(report)与“信件”(letter)文件类的默认页格式. 在这种格式中, 段落之间的距离是固定的, 从而最后一行的位置各不相同. 与此相反的是 `\flushbottom`.

`\raggedleft` 在此命令以后的行只在右边对齐, 左边可以参差不齐, 可用 `\\` 提示换行. 参见 `\begin{flushright}`.

`\raggedright` 在此命令以后的行只在左边对齐, 右边可以参差不齐, 可用 `\\` 提示换行. 参见 `\begin{flushleft}`.

`\raisebox{升高量}[盒高][盒深]{文本}` 把包含文本的 LR 盒子从当前基线提升指定的升高量, 当升高量取负值时则是下降. 当选项盒高与盒深给出时, 则认为此盒子具有指定的高度与深度, 而不管它的实际大小如何.

`\raisetag{长度} [m][a]` 在 $\mathcal{A}\mathcal{M}\mathcal{S}$ -L^AT_EX 里, 本命令把多行公式环境中的公式标号从正常位置提升指定的长度.

`\rangle [m]` 产生 \rangle .

`\rceil [m]` 产生 \rceil .

`\Re [m]` 产生 \Re .

`\ref{标记}` 打印出标记出现处的节号、公式号、图号或表号, 而标记则由命令 `\label{标记}` 设置.

`\reflectbox{文本}` 在输入宏包 `graphics` 后, 本命令使得含文本的 LR 盒子沿中心线作水平翻转(镜射).

`\refname` 在“文章”(article)文件类中含有参考文献标题的命令, 在英语中定义为‘Reference’, 但可被重新定义以适用于其他语言.

`\refstepcounter{计数器}` 类似于 `\stepcounter`, 把计数器中存储的值增加 1, 但本命令也使得计数器成为 `\label-\ref` 交叉参考命令中用到的计数器.

`\renewcommand{命令名}[参量个数][选项]{命令内容}` 类似于 `\newcommand`, 只是 `\命令名` 已经存在, 否则会显示出错信息.

`\renewcommand*{命令名}[参量个数][选项]{命令内容}` 与 `\renewcommand` 类似, 不过命令内容必须是短的, 即在命令内容中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\renewenvironment{环境名}[参量个数][选项]{进入命令}{退出命令}` 类似于 `\newenvironment`, 只是环境名已经存在, 否则会显示出错信息.

`\renewenvironment*{环境名}[参量个数][选项]{进入命令}{退出命令}` 类似于 `\renewenvironment`, 不过 `\begin{环境名}` 的参数必须是短的, 即不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\RequirePackage[选项]{宏包}[版本] [p]` 在类文件或宏包文件内, 本命令等价于 `\usepackage`, 它输入一个或几个以 `sty` 为扩展名的宏包文件, 如果宏包不止一个, 则可用逗号隔开, 选项被应用于所有输入的宏包文件. 此外, 命令 `\documentclass` 中列举的选项也应用于所有输入的宏包文件.

选项版本是形如 `yyyy/mm/dd` 的日期, 例如 `1994/08/01`, 如果宏包文件的日期比它早, 就会显示警告信息.

`\RequirePackageWithOptions{宏包}[版本] [p]` 类似于 `\RequirePackage`, 只是当前使用的所有选项都被应用于输入的宏包中.

`\resizebox{宽度}{高度}{文本}` 在输入宏包 `graphics` 后, 它把包含文本的 LR 盒子放缩到指定的宽度与高度, 如果这两个长度中有一个是 `!`, 则它们使用同一个放大倍数.

`\resizebox*{宽度}{高度}{文本}` 类似于 `\resizebox`, 只是高度是指包括 LR 盒子的高度与深度之和的总高度.

`\reversemarginpar` 把页边注记移到与标准位置 (右侧或‘外’侧) 相反的位置, 可用命令 `\normalmarginpar` 取消其作用.

`\rfloor [m]` 产生 \rfloor .

`\rhd [m]` 产生 \rhd .

`\rho [m]` 产生 ρ .

`\right` 定界符 `[m]` 调整定界符的大小使其适应夹在 `\left ... \right` 内的公式的高度, 例如: `\right]`. 如果另一端没有明显的定界符与之配对, 则可使用空定界符 (`\left.`).

`\Rrightarrow [m]` 产生 \Rightarrow .

- `\rightarrow` [m] 产生 \rightarrow .
- `\rightharpoonup` [m] 产生 \rightarrow .
- `\rightharpoonup` [m] 产生 \rightarrow .
- `\rightleftharpoons` [m] 产生 \rightleftharpoons .
- `\rightmargin` 在 list 环境里, 这是环境内文本的右边界与外部正文的右边界相比的缩进值, 正常值是 0pt, 可用命令 `\setlength` 为其重置新值:
- `\setlength{\rightmargin}{0.5cm}`
- `\rm` 转变为罗马族, 直立形状, 中等粗细系列 (Roman, upright, medium) 的字体属性, 这是默认的字体系.
- `\rmdefault` 本命令定义由 `\rmfamily` 命令选取的族属性, 可用 `\renewcommand` 重新定义:
- `\renewcommand{\rmdefault}{ptm}`
- `\rmfamily` 本命令使字体保持当前的系列与形状属性, 但转变为罗马族属性.
- `\Roman{计数器}` 把计数器的当前值用大写罗马数字打印出来.
- `\roman{计数器}` 把计数器的当前值用小写罗马数字打印出来.
- `\rotatebox{角度}{文本}` 在输入宏包 `graphics` 后, 本命令使得含文本的 LR 盒子以基线的左端点为中心, 按反时针方向旋转指定的角度.
- `\rq` 产生 ' , 同符号 ' .
- `\rule[升高量]{宽度}{高度}` 产生一个具有指定宽度与高度的实心矩形, 当选项升高量给出时, 将此矩形从基线提升指定的升高量. 如果宽度或高度之一取 0pt, 就能得到一条无形的线段, 它能用来产生水平或竖直空白.
- `\rvert` [m][a] 产生 $|$ (右定界符).
- `\rVert` [m][a] 产生 $\|$ (右定界符).
- `\S` 产生 \S .
- `\SS` 产生 $\text{\text{SS}}$.
- `\savebox{\盒子名}[宽度][位置]{文本}` 类似于 `\makebox`, 只是它并不立即打印, 而是保存在 `\盒子名` 中, 这里的 `\盒子名` 必须已经用 `\newsavebox` 定义过. 这个盒子可利用命令 `\usebox{\盒子名}` 被多次使用.
- `\savebox{\子图盒子}(宽度,高度)[位置]{子图图形}` 在 `picture` 环境里, 一个子图图形可被存储于具有指定宽度和高度的 `\子图盒子` 中, 这里的 `\子图盒子` 必须已经用 `\newsavebox` 定义过. 选项位置的意义同 `picture` 环境里的命令 `\makebox`, 这个盒子可利用命令 `\usebox{\盒子名}` 在 `picture` 环境里被多次使用.
- `\sb` 产生一个下标, 同符号 $_$.

`\sbox{盒子名}{文本}` 把文本保存在 LR 盒子 \盒子名 中, 这里的 \盒子名 必须已经用 `\newsavebox` 定义过. 这个盒子可利用命令 `\usebox{盒子名}` 被多次使用.

`\sc` 转变为罗马族, 小型大写形状, 中等粗细系列 (ROMAN, CAPS AND SMALL CAPS, MEDIUM) 的字体属性.

`\scalebox{水平倍数}[竖直倍数]{文本}` 在输入宏包 `graphics` 后, 本命令使得含文本的 LR 盒子分别按水平倍数以及竖直倍数放缩, 当竖直倍数不出现时, 认为它等于水平倍数.

`\scdefault` 本命令定义由 `\scshape` 命令选取的形状属性, 可用 `\renewcommand` 重新定义:

```
\renewcommand{\scdefault}{sc}
```

`\scriptscriptstyle [m]` 在数学模式里使字体大小按 `\scriptscriptstyle` 选取.

`\scriptsize` 把字体尺寸转换成 `\scriptsize`, 它比 `\footnotesize` 小, 比 `\tiny` 大.

`\scriptstyle [m]` 在数学模式里使字体大小按 `\scriptstyle` 选取.

`\scshape` 本声明把字体的形状属性改为小型大写, 但保留族与系列不变.

`\searrow [m]` 产生 \searrow .

`\sec [m]` 本命令生成数学公式里的函数名 ‘sec’.

`\section[短名]{节名}` 开始新的一节, 以当前的章号 (仅出现在 “书籍” 或 “报告” 文件类) 加上一个自动产生的节号, 再添上节名作为节的标题, 如果给出了可选项短名, 那么它将在目录以及书眉上取代原有的节名.

`\section*{节名}` 同 `\section`, 只是没有编号, 而且节名也不出现在目录中.

`\see` 配合 `Makeindex` 程序, 在文件 `makeidx.sty` 中定义的命令, 它出现在 `\index` 命令中, 表示参看索引纪录中的其他条目, 例如:

```
\index{entry|see{reference}}
```

请注意: 在上面例子中 `\` 应被 `|` 取代, 成为 `|see`.

`\seename` 在文件 `makeidx.sty` 中定义的命令, 它含有命令 `\see` 生成的文字, 在英语中定义为 ‘see’, 但可被重新定义以适用于其他语言.

`\selectfont` 激活当前的字体属性, 使它成为当前文本的字体, 它通常出现在改变字体属性的命令后面, 例如:

```
\fontshape{sl}\selectfont
```

`\selectlanguage{语言}` 出现在像 `esperant`, `german` 这样的多种语言包, 或者 `babel` 系统中, 用以改变语种. 其效果是: 标题名、命令 `\today` 所生成的日期格式、与不同语言有关的命令以及单词的拆分模式都发生变化. 例如:

`\selectlanguage{english}`

`\setboolean{开关}{值}` 在已输入 L^AT_EX 标准宏包 `ifthen` 的条件下, 本命令给布尔开关置值, 这里的值必须是 `true` 或 `false`, 而且此开关必须预先用命令 `\newboolean{开关}` 创建. 命令 `\boolean{开关}` 可测试此开关的值, 用于命令 `\ifthenelse` 及 `\whiledo` 的测试部分作为逻辑语句.

`\setcounter{计数器}{值}` 把整数值赋给计数器.

`\setlength{<长度命令>}{长度值}` 把长度值赋给 `<长度命令>`, 这里的长度值可以是固定值或弹性值.

`\SetMathAlphabet{<命令>}{数学变体}{编码}{族}{系列}{形状} [p]` 对于已被 `\DeclareMathAlphabet` 命令声明过的数学模式下设置字体的命令 `<命令>`, 定义特定的数学变体所应取的字体属性, 例如:

`\DeclareMathAlphabet{\mathsl}{bold}{OT1}{cmr}{bx}{sl}`

`\setminus [m]` 产生 `\`.

`\SetSymbolFont{字体}{数学变体}{编码}{族}{系列}{形状} [p]` 对于已被命令 `\DeclareSymbolFont` 声明过的符号字体, 定义特定的数学变体所应取的字体属性.

`\settime{秒}` 在“投影片”(slides)文件类中, 如果选项 `clock` 被选中, 则以分为单位的时间标记将出现在注记的底部, 本命令把内部时钟设置为指定的秒值, 参见 `\addtime`.

`\settodepth{<长度命令>}{文本}` 把文本在基线以下的深度值赋给 `<长度命令>`.

`\settoheight{<长度命令>}{文本}` 把文本在基线以上的高度值赋给 `<长度命令>`.

`\settowidth{<长度命令>}{文本}` 把文本作为 LR 盒子的宽度值赋给 `<长度命令>`.

`\sf` 转变为无衬线族, 直立形状, 中等粗细系列 (sans serif, upright, medium) 的字体属性.

`\sfdefault` 本命令定义由 `\sffamily` 命令选取的族属性, 可用 `\renewcommand` 重新定义:

`\renewcommand{\sfdefault}{phv}`

`\sffamily` 本命令使字体保持当前的系列与形状属性, 但转变为无衬线族属性.

`\sharp [m]` 产生 `#`.

`\shortstack[位置]{文本}` 把文本竖直堆叠成一系列的格式, 其中用 `\\` 指示换行. 选项位置可取值 `l` 或 `r`, 表示文本居左或居右对齐, 默认的是居中, 例如:

`\shortstack{aa\\bbb\\cc} =`

$$\begin{array}{c} \text{aa} \\ \text{bbb} \\ \text{cc} \end{array}$$

`\showhyphens{单词表}` 在终端上显示单词表中各词的可能拆分方式.

`\sideset{前置}{后置}\符号 [m][a]` 把上下标前置与后置分别放在数学\符号的前后, 例如:

$$\sideset{_{\dag^*}}{_{\dag^*}}\prod = \prod_{\dag}^{\dag^*}$$

`\Sigma [m]` 产生 Σ .

`\sigma [m]` 产生 σ .

`\signature{名字}` 用在文件类“信件”(letter)中, 打印在签名的下方, 以给出签名人的名字, 用在签名与`\name`的名字不一致的情形.

`\sim [m]` 产生 \sim .

`\simeq [m]` 产生 \simeq .

`\sin [m]` 本命令生成数学公式里的函数名‘sin’.

`\sinh [m]` 本命令生成数学公式里的函数名‘sinh’.

`\sl` 转变为罗马族, slanted 的斜体形状, 中等粗细系列 (*Roman, slanted, medium*) 的字体属性.

`\sldefault` 本命令定义由`\slshape`命令选取的形状属性, 可用`\renewcommand`重新定义:

`\renewcommand{\sldefault}{sl}`

`\sloppy` 在遇到这个命令后, 词与词之间的间隔允许大于通常的值, 使得一个段落可以分拆成更多的行, 与此相反的命令是`\fussy`, 参见`\begin{sloppypar}`,

`\slshape` 本声明把字体的形状属性改为 *slanted* 的斜体, 但保留族与系列不变.

`\small` 把字体尺寸转换成`\small`, 它比`\normalsize`小, 比`\footnotesize`大.

`\smallskip` 插入一个值为`\smallskipamount`的小的竖直间隔, 参见`\medskip`, `\bigskip`.

`\smallskipamount` 用于`\smallskip`的竖直间隔的标准值, 可用命令`\setlength`加以改变:

`\setlength{\smallskipamount}{1ex plus0.5ex minus0.3ex}`

`\smile [m]` 产生 \smile .

`\sp` 产生一个上标, 同符号 \wedge .

`\spadesuit [m]` 产生 \spadesuit .

`\sqcap [m]` 产生 \sqcap .

`\sqcup [m]` 产生 \sqcup .

`\sqrt[n]{变量} [m]` 本命令产生一个根号, 可选参数 n 是根指数, 如`\sqrt[3]{2}`
 $= \sqrt[3]{2}$, `\sqrt{2}` $= \sqrt{2}$.

`\sqsubset [m]` 产生 \sqsubset .

`\sqsubseteq [m]` 产生 \sqsubseteq .

`\sqsupset [m]` 产生 \sqsupset .

`\sqsupseteq [m]` 产生 \sqsupseteq .

`\ss` 产生 β .

`\stackrel{上部}{下部} [m]` 把数学符号上部放在符号下部的上面, 而且使上部用较小的字体打印:

$$\stackrel{\alpha}{\longrightarrow}$$

`\star [m]` 产生 \star .

`\stepcounter{计数器}` 使计数器的数增加 1.

`\stretch{十进小数}` 是一个弹性长度, 其自然值是 0pt, 其伸展性等于十进小数乘以 `\fill`.

`\subitem{子条目}` 在 `theindex` 环境里, 本命令生成低于 `\item` 的一个二级子条目.

`\subjectname` 这是“信件”(letter)文件类的命令, 它打印出命令 `\subject` 的文字, 在英语环境里是‘Subject’, 但可被重新定义以适应其他语言.

`\subparagraph[短标题]{标题}` 章节层次中的最后一级, 位于 `\paragraph` 之下, 它在当前段落编号后面添加自动生成的小段序号, 后面再打印标题. 可选项短标题用在目录中代替标题.

`\subparagraph*{标题}` 类似于 `\subparagraph`, 只是没有编号, 也不出现在目录里.

`\subsection[短标题]{标题}` 章节层次中介于 `\section` 与 `\subsubsection` 之间, 它在当前节号的后面添加自动生成的小节序号, 后面再打印标题. 可选项短标题用在目录中代替标题.

`\subsection*{标题}` 类似于 `\subsection`, 只是没有编号, 也不在目录中出现.

`\substack{第一行\\...\\最后一行} [m][a]` 生成居中对齐的多行上下标或上下界, 它被括号 `{..}` 扩起来后紧跟在 `^` 或 `_` 的后面.

`\subsubitem{子条目}` 在 `theindex` 环境里, 本命令生成低于 `\subitem` 的一个三级子条目.

`\subsubsection[短标题]{标题}` 章节层次中介于 `\subsection` 与 `\paragraph` 之间, 它在当前小节号的后面添加自动生成的次小节序号, 后面再打印标题. 可选项短标题用在目录中代替标题.

`\subsubsection*{标题}` 类似于 `\subsubsection`, 只是没有编号, 也不在目录中出现.

`\subset [m]` 产生 \subset .

`\subseteq [m]` 产生 \subseteq .

- `\succ [m]` 产生 \succ .
- `\succeq [m]` 产生 \succeq .
- `\sum [m]` 产生 \sum .
- `\sup [m]` 在数学公式中产生函数名 ‘sup’ 的命令, 可带下标作为下界.
- `\suppressfloats[位置]` 从这个命令至当前页结尾之间的浮动单元都被推迟到下一页及以后, 选项位置可取值 t 或 b (只能取一个), 表示只推迟符合此选项的浮动单元.
- `\supset [m]` 产生 \supset .
- `\supseteq [m]` 产生 \supseteq .
- `\surd [m]` 产生 \surd .
- `\swarrow [m]` 产生 \swarrow .
- `\symbol{n}` 产生当前字体中内部编号为 n 的字符.
- `\t{xy}` 产生连接两个字符的重音号: $\t{oo} = \text{\textcircled{O}}$.
- `\tabbingsep` 在 `tabbing` 环境里, 如果遇到文本 `\`, 命令 `\` 表示跳到当前列的末尾, 使文本右对齐, 这时 `\tabbingsep` 规定了文本与右边制表位间的留空. 可用 `\setlength` 为其重置新值.
- `\tabcolsep` 在 `tabular` 环境里, 两列文本间的留空等于 `\tabcolsep` 的两倍. 可用 `\setlength` 为其重置新值:
- `\setlength{\tabcolsep}{3mm}`
- `\tableofcontents` 把目录打印出来.
- `\tablename` 含有表格标题的命令, 在英语中定义为 ‘Table’, 但可被重新定义以适用于其他语言.
- `\tabularnewline[长度]` 在 `tabular` 或 `array` 环境里, 结束当前行, 如选项长度被给出, 则再插入指定长度的竖直间隔. 这等价于 `\\[长度]`, 不过本命令的意义更明确, 不会造成究竟是一列内部的换行还是整行的换行的含糊不清的情形. 尤其在最后一列中出现命令 `\raggedright` 时, 这时必须用 `\tabularnewline` 代替 `\\`.
- `\tag{标记} [m][a]` 在 `AMS-LATEX` 的多行公式环境里, 本命令打印出加上括号的 (标记) 以取代公式编号, 本命令带 * 号的形式不加括号.
- `\tan [m]` 本命令生成数学公式里的函数名 ‘tan’.
- `\tanh [m]` 本命令生成数学公式里的函数名 ‘tanh’.
- `\tau [m]` 产生 τ .
- `\tbinom{上部}{下部} [m][a]` 类似于 `\binom`, 产生组合数, 不过由 `\textstyle` 确定字体大小.

`\telephone{数字}` 用于“信件”(letter)文件类中,输入发信人的电话号码.在标准 L^AT_EX 的 letter.sty 中,仅当 `\address` 命令不被调用时才会打印这个数字.本命令的目的是为了公司信件格式如 mpletter 的需要.

`\TeX` 产生 T_EX.

`\text{短文本}` [m][a] 在输入宏包 amsmath 或 amstext 后,本命令在数学环境里按正文格式打印短文本,如果出现在上下标,它会自动调节字体的大小.

`\text` 符号名 本系列命令是为了产生一些原本要用数学模式或字母组合才能得到的符号,它们有:

```
\textbullet (●); \textemdash (—); \textendash (-);
\textexclamdown (¡); \textperiodcentered (·);
\textquestiondown (¿); \textquotedblleft (“);
\textquotedblright (”); \textquoteleft (‘);
\textquoteright (’); \textvisiblespace (␣);
\textasciicircum (^); \textasciitilde (~);
\textbackslash (\); \textbar (|);
\textgreater (>); \textless (<).
```

`\textbf{文本}` 本命令等价于 `\bfseries 文本`,它使用当前族与形状,但是黑体系列(bold)的字体来打印文本.

`\textcircled{字符}` 在字符外面加一个圈: `\textcircled{s} = \textcircled{s}`.

`\textcolor 色彩{文本}` 宏包 color 中定义的命令,它把文本用指定的色彩打印,色彩的定义方法同 `\color`.

`\textcompwordmark` 本命令使两个字母不产生连字,如:

```
f\textcompwordmark i = fi
```

`\textfloatsep` 位于顶部或底部的浮动单元与相邻的正文之间的间隔距离,可用 `\setlength` 命令为其重置新值:

```
\setlength{\textfloatsep}{20pt plus2pt minus4pt}
```

`\textfraction` 在正文与浮动单元混合的页面里,正文所占的最小比例,其值是一个十进小数.可用以下命令为其重置新值:

```
\renewcommand{\textfraction}{0.5}
```

`\textheight` 每页正文所占的高度,不包括页眉和页底的页码,可用 `\setlength` 命令为其重置新值:

```
\setlength{\textheight}{45\baselineskip}
```

`\textit{文本}` 本命令等价于 `\itshape 文本`,它使用当前族与系列,但是斜体(*italic*)形状的字体来打印文本.

`\textmd{文本}` 本命令等价于 `{\mdseries 文本}`, 它使用当前族与形状, 但是中等权重 (medium) 系列的字体来打印文本.

`\textnormal{文本}` 本命令等价于 `{\normalfont 文本}`, 它使用默认的族、系列与形状的字体来打印文本.

`\textquotedbl` 当 T1 编码被激活时, 产生字母 ".

`\textregistered` 产生 ®.

`\textrm{文本}` 本命令等价于 `{\rmfamily 文本}`, 它使用当前系列与形状, 但是罗马族的字体来打印文本.

`\textsc{文本}` 本命令等价于 `{\scshape 文本}`, 它使用当前族与系列, 但是小型大写 (CAPS AND SMALL CAPS) 形状的字体来打印文本.

`\textsf{文本}` 本命令等价于 `{\sffamily 文本}`, 它使用当前系列与形状, 但是无衬线 (sans serif) 族的字体来打印文本.

`\textsl{文本}` 本命令等价于 `{\slshape 文本}`, 它使用当前族与系列, 但 *slanted* 斜体形状的字体来打印文本.

`\textstyle [m]` 在数学模式里使字体大小按 `\textstyle` 选取.

`字符` 在当前文本里产生一个上标, 例如:

`ab = ab.`

`\texttrademark` 产生 ™.

`\texttt{文本}` 本命令等价于 `{\ttfamily 文本}`, 它使用当前系列与形状, 但是打字机 (typewriter) 族的字体来打印文本.

`\textup{文本}` 本命令等价于 `{\upshape 文本}`, 它使用当前族与系列, 但是直立 (upright) 形状的字体来打印文本.

`\textwidth` 页面中正文的宽度, 在双栏格式中, 这个宽度等于两个栏的宽度加上中间的留空. 可用 `\setlength` 为其重置新值.

`\tfrac{分子}{分母} [m][a]` 按 `\textstyle` 的字体大小产生一个分式.

`\TH` 当 T1 编码被激活时, 产生字母 Þ.

`\th` 当 T1 编码被激活时, 产生字母 þ.

`\thanks{脚注文本}` 当遇到命令 `\maketitle` 时, 在标题页内针对作者名产生一个内容为脚注文本的脚注.

`\the` 计数器 把计数器的数格式化后打印的内部命令, 其输出可供其他计数器使用, 例如, `\thesubsection` 可被定义为

`\thesection.\roman{subsection}`

可用 `\renewcommand{\the计数器}{定义}` 重新定义.

`\Theta [m]` 产生 Θ.

`\theta` [m] 产生 θ .

`\thicklines` 在 `picture` 环境里, 此命令增加斜线、箭头、圆以及圆角矩形的线条厚度, 使它们比正常情形更厚.

`\thinlines` 在 `picture` 环境里, 此命令重置斜线、箭头、圆以及圆角矩形的线条厚度, 使它们回复正常.

`\thinspace` [a] 这是 `\,` 的别名, 在数学模式里产生一个小的空白.

`\thispagestyle{格式}` 本命令只改变当前页的格式, 格式的取值可以是: `plain`, `empty`, `headings` 以及 `myheadings`.

`\tilde{x}` [m] 产生数学变量上的重音号: `\tilde{a} = \tilde{a}`.

`\Tilde{x}` [m][a] 本命令与 `\Tilde` 同样使用, 不过当出现 \mathcal{AMS} -L^AT_EX 的多重数学重音号时, 本命令会自动调节位置.

`\tiny` 把字体尺寸转换成最小的 `\tiny`, 它比 `\scriptsize` 小.

`\times` [m] 产生 \times .

`\title{文本}` 遇到 `\maketitle` 命令后, 在标题页产生内容为文本的标题.

`\rightarrow` [m] 产生 \rightarrow .

`\today` 以美国方式打印当前日期, 如要改成英国式或其他语言, 则可利用 `\day`, `\month` 以及 `\year` 对其重新定义.

`\top` [m] 产生 \top .

`\topfigrule` 本命令仅当一个浮动环境出现在页顶时被执行, 它通常被定义为空, 但可被重定义以在页面的正文与浮动部分之间画一条直线, 请注意这条命令不应该产生额外的竖直空间, 例如:

```
\renewcommand{\topfigrule}{\vspace*{-.4pt}
\rule{\columnwidth}{.4pt}}
```

`\topfraction` 页面中可供顶部的浮动单元使用的最大部分, 这是一个十进小数, 可用以下命令重置其值:

```
\renewcommand{\topfraction}{0.5}
```

`\topmargin` 从纸顶到页眉的距离, 可用 `\setlength` 为其重置新值:

```
\setlength{\topmargin}{0.5in}
```

`\topnumber` 可出现在一个页面顶部的浮动单元最大个数, 可用命令

```
\setcounter{topnumber}{3}
```

为其重置新值.

`\topsep` 在 `list` 环境的头尾, 除了 `\parskip` 外, `\topsep` 规定了额外的竖直空白. 当文件类选项 `fleqn` 被选取后, 在行间数学公式的前后也被插入这样的空白. 可用 `\setlength` 为其重置新值:

`\setlength{\topsep}{4pt plus2pt minus2pt}`

`\topskip` 从正文的顶部到正文第一行基线的距离, 可用 `\setlength` 为其重置新值:

`\setlength{\topskip}{12pt}`

`\totalheight` 一个盒子的总高度, 即高度与深度之和, 用于 `\makebox`, `\framebox` 以及 `\savebox` 的宽度参数中, 或者 `\parbox` 及 `minipage` 环境的高度参数中, 如:

`\framebox[6\totalheight]{text}`

`totalnumber` 可出现在一个页面里的浮动单元最大个数, 可用命令

`\setcounter{totalnumber}{3}`

为其重置新值.

`\triangle [m]` 产生 \triangle .

`\triangleleft [m]` 产生 \triangleleft .

`\triangleright [m]` 产生 \triangleright .

`\tt` 转变为打字机族, 直立形状, 中等粗细系列 (`typewriter`, `upright`, `medium`) 的字体属性.

`\ttdefault` 本命令定义由 `\ttfamily` 命令选取的族属性, 可用 `\renewcommand` 重新定义:

`\renewcommand{\ttdefault}{pcr}`

`\ttfamily` 使字体保持当前的系列与形状属性, 但转变为打字机 (`typewriter`) 族属性.

`\twocolumn[文本]` 另起一页并转换成双栏格式, 选项文本被排成一栏, 跨越两栏的分隔.

`\typein[命令]{信息}` 在终端屏幕上显示信息, 并等待用户输入回答, 用户回答的文本被放入选项 `命令` 中, 或在没有选项时放入 `\@typein` 内, 等用户输入回车后程序继续进行.

`\typeout{信息}` 把信息输出在终端屏幕上以及 `log` 文件中, 同时程序继续进行.

`\u{x}` 产生一个重音号: `\u{o}=ö`.

`\unboldmath` 与命令 `\boldmath` 的作用相反, 它必须在进入数学模式前出现在文本模式中, 使得数学模式的字体恢复正常, 即数学斜体.

`\underbrace{公式} [m]` 在公式的下面产生一个水平的花括号, 以下标形式出现在本命令后面的数学符号将被居中放在花括号的下方. 例如:

`\underbrace{a+b}=a+b`

$$\underbrace{x+y+z}_{\xi\eta\zeta} = x + y + z$$

`\underleftarrow{公式}` [m][a] 在公式的下面画一条指向左方的长箭头.

`\underleftrightharrow{公式}` [m][a] 在公式的下面画一条左右两边都有箭头的水平线.

`\underline{文本}` 不论数学模式还是文本模式, 都在文本的下面产生一条水平直线:

`\underline{Text} = \underline{Text}`

`\underrightarrow{公式}` [m][a] 在公式的下面画一条指向右方的长箭头.

`\underset{字符}{\符号}` [m][a] 把字符按下标的字体放在\符号下方:

$$\underset{\beta}{\longrightarrow}$$

`\unitlength` 为下面的 `picture` 环境定义长度单位, 可用 `\setlength` 为其重置新值:

`\setlength{\unitlength}{1mm}`

`\unlhd` [m] 产生 \leq .

`\unrhd` [m] 产生 \geq .

`\Uparrow` [m] 产生 \Uparrow .

`\uparrow` [m] 产生 \uparrow .

`\updefault` 本命令定义了由命令 `\upshape` 选取的直立形状字体, 可用以下命令予以重新定义:

`\renewcommand{\updefault}{n}`

`\Updownarrow` [m] 产生 \Updownarrow .

`\updownarrow` [m] 产生 \updownarrow .

`\uplus` [m] 产生 \uplus .

`\upshape` 本声明把字体的形状属性改为直立, 但保留族与系列不变.

`\Upsilon` [m] 产生 Υ .

`\upsilon` [m] 产生 υ .

`\usebox{\盒子名}` 把\盒子名中包含的内容插入当前文本, 以此为名的盒子是由前面的命令 `\newsavebox` 创建的, 用命令 `\sbox` 或 `\savebox` 存储的.

`\usecounter{计数器}` 这是 `list` 环境里的命令, 它为 `\item` 命令指定一个计数器, 每次调用 `\item` 都会使这个计数器的值增加 1.

`\usefont{编码}{族}{系列}{形状}` 激活具有指定属性, 但是大小保持当前值的字体, 本命令等价于先选定字体属性, 然后执行命令 `\selectfont`.

`\usepackage`[选项表]{宏包}[版本] [p] 输入一个或几个含有附加 L^AT_EX 或 T_EX 命令的宏包文件, 文件的扩展名是 sty, 如有多个文件, 则用逗号隔开. 选项表被应用于所有的宏包, 此外, `\documentclass` 中指定的选项也被应用于各个宏包.

选项版本是格式为 yyyy/mm/dd 的日期, 例如 1994/08/01. 当输入的宏包文件早于此日期时会显示警告信息. 例如:

```
\usepackage{bezier,ifthen}[1994/06/01]
```

`\v{x}` 产生一个重音号: `\v{o}` = ö.

`\value{计数器}` 输出存储在计数器里的数值供其他命令使用, 但它并非打印一个数. 例如命令 `\setcounter{计数器1}{\value{计数器2}}` 把计数器 2 的值赋给计数器 1.

`\varepsilon` [m] 产生 ε .

`\varinjlim` [m][a] 在公式里产生 \varinjlim .

`\varliminf` [m][a] 在公式里产生 \varliminf .

`\varlimsup` [m][a] 在公式里产生 \varlimsup .

`\varphi` [m] 产生 φ .

`\varpi` [m] 产生 ϖ .

`\varprojlim` [m][a] 在公式里产生 \varprojlim .

`\varrho` [m] 产生 ϱ .

`\varsigma` [m] 产生 ς .

`\vartheta` [m] 产生 ϑ .

`\vdash` [m] 产生 \vdash .

`\vdots` [m] 产生 \vdots .

`\vec{x}` [m] 在变量 x 上加一个箭头: `\vec{a}` = \vec{a} .

`\vector(\Delta x, \Delta y){长度}` 这是 picture 环境里的一个图形元素, 通常作为 `\put` 或 `\multiput` 的参数, 其作用是绘制水平或竖直向量以及某些特定斜率的向量. 对于水平或竖直向量, 长度就是向量的实际长度 (以 `\unitlength` 为单位), 对于斜向量, 长度是它在 x 轴上的投影 (即水平位移). $(\Delta x, \Delta y)$ 给出了向量的斜率, 其中 Δx 与 Δy 是一对互素整数, 满足 $-4 \leq \Delta x \leq 4$, $-4 \leq \Delta y \leq 4$.

`\vee` [m] 产生 \vee .

`\verb|文本|` 在两根竖线 `|...|` 中间的内容都被不加改动地用打字机字体打印, 特殊符号以及命令都被当成普通字符输出. 除星号 * 以外的字符都可以取代竖线 | 作为开关字符, 唯一的条件是文本内不能含有此字符.

`\verb*|文本|` 类似于 `\verb`, 不过空格被打印成 `_`.

`\vfill` 竖直弹性长度, 其自然长度等于 0, 但可伸展成任意值, 它可用于以空白填满页面的一部分, 实际上这个命令是 `\vspace{\fill}` 的缩写.

`\visible` 当文件类是“投影片”(slides)时, 本命令的作用与 `\invisible` 相反, 使得文本被正常打印. 它的作用范围可持续到所在局部环境的结束, 或遇到 `\invisible` 命令, 本命令可用于生成覆盖片.

`\vline` 在 tabular 环境里, 本命令在表格的列条目内打印一条竖直线.

`\voffset` 输出页面与打印机设定的打印起始线之间的竖直位移, 通常打印起始线与纸顶的距离是 1in, `\voffset` 的标准值是 0pt, 可以用 `\setlength` 命令重置新值:

```
\setlength{\voffset}{-1in}
```

`\vspace{高度}` 产生具有指定高度的竖直空白, 当它位于页首或页尾时则被忽略.

`\hspace*{高度}` 产生具有指定高度的竖直空白, 即使位于页首或页尾时也不被忽略.

`\wedge [m]` 产生 \wedge .

`\whiledo{测试}{命令序列}` 输入宏包 ifthen 后才有效的命令, 当逻辑语句测试的值为真时, 命令序列被反复执行. 逻辑语句可以是: 关系式(用 $< = >$ 连接的两个数), 奇偶性测试(`\isodd{整数}`), 两个文本的比较(`\equal{文本一}{文本二}`), 两个长度的比较(`\lengthtest{长度一 op 长度二}`, 其中 op 是 $< = >$ 中之一)或布尔开关的测试(`\boolean{开关}`), 这里的布尔开关是由命令 `\newboolean{开关}` 创立的, 并且由命令 `\setboolean{开关}{值}` 置值, 其中的值可以为 true 或 false, 逻辑语句还可以用逻辑算子 `\and`, `\or` 以及 `\not` 联合, 并用 \backslash 与 \backslash 分组.

测试 `\isodd`, `\lengthtest` 以及 `\boolean` 是 L^AT_EX 2_ε 的新内容.

`\widehat{xyz} [m]` 在几个符号的上面打印一个加宽的 `\hat`:

```
\widehat{xyz} = \widehat{xyz}
```

`\widetilde{xyz} [m]` 在几个符号的上面打印一个加宽的 `\tilde`:

```
\widetilde{xyz} = \widetilde{xyz}
```

`\width` 等于一个盒子的自然宽度的长度参数, 它主要用于 `\makebox`, `\framebox` 或 `\savebox` 命令的宽度参数内, 或者用于 `\parbox` 及 `minipage` 环境的高度参数中, 例如:

```
\framebox[2\width]{text}
```

`\wp [m]` 产生 \wp .

`\wr [m]` 产生 \wr .

`\Xi [m]` 产生 Ξ .

`\xi [m]` 产生 ξ .

`\xleftarrow[下部]{上部} [m][a]` 画一个指向左方的箭头, 其上方用上标字体打印上部, 其下方用下标字体打印选项下部.

`\xrightarrow[下部]{上部} [m][a]` 画一个指向右方的箭头, 其上方用上标字体打印上部, 其下方用下标字体打印选项下部.

`\zeta [m]` 产生 ζ .

附录三 字体表

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	ff 11	fi 12	fl 13	ffi 14	ffl 15
'02x	ı 16	ı 17	` 18	´ 19	˘ 20	˙ 21	˚ 22	˛ 23
'03x	ˆ 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	˜ 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	i 60	= 61	¿ 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	˘ 95
'14x	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	- 123	— 124	" 125	~ 126	¨ 127

字体布局 1: cmr10. OT1 编码方案中字符与编码数值的标准对应关系.

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	ff 11	fi 12	fl 13	ffi 14	ffl 15
'02x	ı 16	ı 17	` 18	´ 19	˘ 20	˙ 21	˚ 22	˛ 23
'03x	ˆ 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	˜ 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	i 60	= 61	¿ 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	˘ 95
'14x	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	- 123	— 124	" 125	~ 126	¨ 127

字体布局 2: cmss10. OT1 编码方案. 本字体集是无衬线字体.

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Τ 7
'01x	Φ 8	Ψ 9	Ω 10	↑ 11	↓ 12	' 13	ı 14	¿ 15
'02x	ı 16	J 17	` 18	˘ 19	˙ 20	˚ 21	˛ 22	˜ 23
'03x	ˆ 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	ı 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	\ 92] 93	^ 94	_ 95
'14x	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	{ 123	124	} 125	~ 126	¨ 127

字体布局 3: cmtt10. 本字体集是 typewriter (打字机) 字体. 与字体布局 1 的不同之处是位于 11-15, 60, 62, 92, 123, 124 和 125 号位置的符号. 对于 *italic typewriter* 字体, 还用 £ 取代 \$.

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Τ 7
'01x	Φ 8	Ψ 9	Ω 10	↑ 11	↓ 12	' 13	ı 14	¿ 15
'02x	ı 16	J 17	` 18	˘ 19	˙ 20	˚ 21	˛ 22	˜ 23
'03x	ˆ 24	SS 25	Æ 26	Œ 27	Ø 28	Æ 29	Œ 30	Ø 31
'04x	ˆ 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	˙ 95
'14x	` 96	A 97	B 98	C 99	D 100	E 101	F 102	G 103
'15x	H 104	I 105	J 106	K 107	L 108	M 109	N 110	O 111
'16x	P 112	Q 113	R 114	S 115	T 116	U 117	V 118	W 119
'17x	X 120	Y 121	Z 122	- 123	— 124	" 125	~ 126	¨ 127

字体布局 4: cmcsc10. 本字体集是 SMALL CAPS 字形. 与字体布局 1 的不同之处是位于 11-15, 25, 60 和 62 号位置的符号. 97-122 号位置是小型大写字母.

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	ff 11	fi 12	fl 13	ffi 14	fff 15
'02x	ı 16	Ŷ 17	` 18	´ 19	˘ 20	˙ 21	˚ 22	° 23
'03x	˘ 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	˘ 32	! 33	" 34	# 35	£ 36	% 37	€ 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	i 60	= 61	j 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	· 95
'14x	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	- 123	— 124	" 125	~ 126	¨ 127

字体布局 5: cmti10. 本字体集是 *italic* 字形. 与字体布局 1 的不同之处是位于 36 号位置的符号(用 £ 取代了 \$).

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	ff 11	fi 12	fl 13	ffi 14	fff 15
'02x	ı 16	Ŷ 17	` 18	´ 19	˘ 20	˙ 21	˚ 22	° 23
'03x	˘ 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	˘ 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	i 60	= 61	j 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	· 95
'14x	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	- 123	— 124	" 125	~ 126	¨ 127

字体布局 6: cmsl10. 本字体集是 *slanted* 字形.

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	α 11	β 12	γ 13	δ 14	ϵ 15
'02x	ζ 16	η 17	θ 18	ι 19	κ 20	λ 21	μ 22	ν 23
'03x	ξ 24	π 25	ρ 26	σ 27	τ 28	υ 29	ϕ 30	χ 31
'04x	ψ 32	ω 33	ε 34	ϑ 35	ϖ 36	ϱ 37	ς 38	φ 39
'05x	\leftarrow 40	\rightharpoonup 41	\rightarrow 42	\dashrightarrow 43	\prime 44	'° 45	\triangleright 46	\triangleleft 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	\cdot 58	$,$ 59	$<$ 60	$/$ 61	$>$ 62	\star 63
'10x	∂ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	b 91	t 92	\sharp 93	\smile 94	\frown 95
'14x	ℓ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	i 123	j 124	\wp 125	\rightarrow 126	\sim 127

字体布局 7: **cmmt10**. OML 编码方案. 在位置 0–39 是希腊字母. 在位置 40–47, 60–64 和 123–127 是数学符号. 粗体版本 **cmmtb10** 具有同样的编码.

	0	1	2	3	4	5	6	7
'00x	$-$ 0	\cdot 1	\times 2	$*$ 3	\div 4	\diamond 5	\pm 6	\mp 7
'01x	\oplus 8	\ominus 9	\otimes 10	\oslash 11	\odot 12	\bigcirc 13	\circ 14	\bullet 15
'02x	\asymp 16	\equiv 17	\subseteq 18	\supseteq 19	\leq 20	\geq 21	\preceq 22	\succeq 23
'03x	\sim 24	\approx 25	\subset 26	\supset 27	\ll 28	\gg 29	\prec 30	\succ 31
'04x	\leftarrow 32	\rightarrow 33	\uparrow 34	\downarrow 35	\leftrightarrow 36	\nearrow 37	\searrow 38	\approx 39
'05x	\Leftarrow 40	\Rightarrow 41	\Uparrow 42	\Downarrow 43	\Leftrightarrow 44	\nwarrow 45	\nearrow 46	\propto 47
'06x	\prime 48	∞ 49	\in 50	\ni 51	Δ 52	∇ 53	$/$ 54	\prime 55
'07x	\forall 56	\exists 57	\neg 58	\emptyset 59	\Re 60	\Im 61	\top 62	\perp 63
'10x	\aleph 64	\mathcal{A} 65	\mathcal{B} 66	\mathcal{C} 67	\mathcal{D} 68	\mathcal{E} 69	\mathcal{F} 70	\mathcal{G} 71
'11x	\mathcal{H} 72	\mathcal{I} 73	\mathcal{J} 74	\mathcal{K} 75	\mathcal{L} 76	\mathcal{M} 77	\mathcal{N} 78	\mathcal{O} 79
'12x	\mathcal{P} 80	\mathcal{Q} 81	\mathcal{R} 82	\mathcal{S} 83	\mathcal{T} 84	\mathcal{U} 85	\mathcal{V} 86	\mathcal{W} 87
'13x	\mathcal{X} 88	\mathcal{Y} 89	\mathcal{Z} 90	\cup 91	\cap 92	\oplus 93	\wedge 94	\vee 95
'14x	\vdash 96	\dashv 97	\lfloor 98	\rfloor 99	\lceil 100	\rceil 101	$\{$ 102	$\}$ 103
'15x	\langle 104	\rangle 105	$\ $ 106	\parallel 107	\updownarrow 108	\updownarrow 109	\backslash 110	\wr 111
'16x	\surd 112	Π 113	∇ 114	\int 115	\sqcup 116	\sqcap 117	\sqsubseteq 118	\sqsupseteq 119
'17x	\S 120	\dagger 121	\ddagger 122	\P 123	\clubsuit 124	\diamond 125	\heartsuit 126	\spadesuit 127

字体布局 8: **cmsy10**. OMS 编码方案. 除了包含许多数学符号之外, 在位置 65–90 是 calligraphic (手写花体) 字母. 粗体版本 **cmsyb10** 具有同样的编码.

	0	1	2	3	4	5	6	7
'00x	(0)	1	[2]	3	4	5	6	7
'01x	{ 8 }	9	< 10	> 11	12	13	/ 14	\ 15
'02x	(16)	17	(18)	19	[20]	21	22	23
'03x	[24]	25	{ 26 }	} 27	< 28	> 29	/ 30	\ 31
'04x	(32)	33	[34]] 35	36	37	[38]] 39
'05x	{ 40 }	41	< 42	> 43	/ 44	\ 45	/ 46	\ 47
'06x	(48)) 49	[50]] 51	52	53	54	55
'07x	(56)) 57	(58)) 59	{ 60 }	} 61	' 62	63
'10x	\ 64	/ 65	66	67	< 68	> 69	□ 70	□ 71
'11x	§ 72	§ 73	⊙ 74	⊙ 75	⊕ 76	⊕ 77	⊗ 78	⊗ 79
'12x	Σ 80	Π 81	∫ 82	U 83	∩ 84	⊕ 85	Λ 86	V 87
'13x	Σ 88	Π 89	∫ 90	U 91	∩ 92	⊕ 93	Λ 94	V 95
'14x	Π 96	Π 97	~ 98	~ 99	~ 100	~ 101	~ 102	~ 103
'15x	[104]] 105	106	107	[108]] 109	{ 110 }	} 111
'16x	√ 112	√ 113	√ 114	√ 115	√ 116	117	┐ 118	119
'17x	↑ 120	↓ 121	↶ 122	↷ 123	↶ 124	↷ 125	↶ 126	↷ 127

字体布局 9: cmex10. OMX 编码方案. 包含具有可变尺寸的数学符号.

	0	1	2	3	4	5	6	7
'00x	0	1	2	3	4	5	6	7
'01x	{ 8	} 9	{ 10	} 11	{ 12	} 13	{ 14	} 15
'02x	16	17	18	19	20	21	22	23
'03x	← 24	↖ 25	→ 26	↗ 27	28	29	30	31
'04x	← 32	→ 33	↑ 34	↓ 35	↔ 36	↗ 37	↘ 38	39
'05x	⇐ 40	⇒ 41	↑ 42	↓ 43	⇔ 44	↖ 45	↘ 46	47
'06x	48	∞ 49	50	51	52	53	54	55
'07x	(56) 57	(58) 59	{ 60	} 61	' 62	63
'10x	64	65	66	67	68	69	70	71
'11x	¢ 72	¢ 73	74	75	76	77	78	79
'12x	Σ 80	Π 81	∫ 82	83	84	85	86	87
'13x	Σ 88	Π 89	∫ 90	91	92	93	94	95
'14x	Π 96	Π 97	98	99	100	101	102	103
'15x	104	105	106	107	↕ 108	↕ 109	110	111
'16x	112	113	114	115	116	117	118	119
'17x	120	121	↵ 122	↵ 123	↵ 124	↵ 125	126	127

字体布局 10: euex10. 美国数学会设计的一种字体. 注意积分号是直立的.

	0	1	2	3	4	5	6	7
'00x	Ѓ 0	Ѕ 1	Ї 2	Э 3	Ї 4	Є 5	Ђ 6	Ѓ 7
'01x	Ѓ 8	Ѕ 9	Ї 10	Э 11	Ї 12	Є 13	Ђ 14	Ѓ 15
'02x	Ю 16	Ж 17	Й 18	Ё 19	Ў 20	Ө 21	Ѕ 22	Я 23
'03x	ю 24	ж 25	й 26	ё 27	ў 28	ө 29	ѕ 30	я 31
'04x	“ 32	! 33	” 34	Ђ 35	Ѓ 36	Ѕ 37	’ 38	’ 39
'05x	(40) 41	* 42	Ђ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	« 60	1 61	» 62	? 63
'10x	˘ 64	А 65	Б 66	В 67	Д 68	Е 69	Ф 70	Г 71
'11x	Х 72	И 73	Ј 74	К 75	Л 76	М 77	Н 78	О 79
'12x	П 80	Ч 81	Р 82	С 83	Т 84	У 85	В 86	Ш 87
'13x	Ш 88	Ы 89	З 90	[91	“ 92] 93	Ь 94	Ъ 95
'14x	‘ 96	а 97	б 98	ц 99	д 100	е 101	ф 102	г 103
'15x	х 104	и 105	ј 106	к 107	л 108	м 109	н 110	о 111
'16x	п 112	ч 113	р 114	с 115	т 116	у 117	в 118	ш 119
'17x	ш 120	ы 121	з 122	— 123	— 124	№ 125	ь 126	ъ 127

字体布局 11: wncyr10. OT2 编码方案. 华盛顿大学的西里尔字体之一.

	0	1	2	3	4	5	6	7
'00x	` 0	´ 1	^ 2	~ 3	¨ 4	˘ 5	° 6	˘ 7
'01x	˘ 8	˘ 9	˘ 10	˘ 11	˘ 12	˘ 13	˘ 14	˘ 15
'02x	“ 16	” 17	„ 18	« 19	» 20	— 21	— 22	— 23
'03x	o 24	l 25	j 26	ff 27	fi 28	fi 29	ffi 30	ffi 31
'04x	˘ 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	\ 92] 93	^ 94	_ 95
'14x	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	{ 123	124	} 125	~ 126	- 127
'20x	À 128	Á 129	Â 130	Ã 131	D 132	E 133	Ê 134	Ë 135
'21x	Ì 136	Í 137	Î 138	Ñ 139	N 140	Ð 141	Ò 142	Ó 143
'22x	Ô 144	Õ 145	Ö 146	Š 147	T 148	Ŧ 149	Û 150	Ü 151
'23x	Ý 152	Ž 153	Ž 154	Ž 155	IJ 156	İ 157	đ 158	š 159
'24x	ă 160	ą 161	ć 162	č 163	ď 164	ě 165	ę 166	ğ 167
'25x	í 168	ï 169	ı 170	ñ 171	ň 172	ŋ 173	ō 174	ř 175
'26x	ř 176	ś 177	š 178	ş 179	ţ 180	ţ 181	ű 182	ű 183
'27x	ÿ 184	ž 185	ž 186	ž 187	ij 188	i 189	l 190	l 191
'30x	Ä 192	Å 193	Ä 194	Ä 195	Ä 196	Ä 197	Æ 198	Ç 199
'31x	È 200	É 201	Ê 202	Ë 203	Ì 204	Í 205	Î 206	Ï 207
'32x	Ð 208	Ñ 209	Ò 210	Ó 211	Ô 212	Õ 213	Ö 214	Œ 215
'33x	Ø 216	Û 217	Ü 218	Ü 219	Ü 220	Ý 221	Þ 222	ŠS 223
'34x	à 224	á 225	â 226	ã 227	ä 228	å 229	æ 230	ç 231
'35x	è 232	é 233	ê 234	ë 235	ì 236	í 237	î 238	ï 239
'36x	ð 240	ñ 241	ò 242	ó 243	ô 244	õ 245	ö 246	œ 247
'37x	ø 248	ù 249	ú 250	û 251	ü 252	ý 253	þ 254	ß 255

字体布局 12: ecrm1000. T1 编码方案.

	0	1	2	3	4	5	6	7
'00x	˘ 0	˙ 1	˚ 2	˛ 3	˜ 4	˝ 5	˚ 6	˘ 7
'01x	˘ 8	˙ 9	˚ 10	˛ 11	˜ 12	˝ 13	˚ 14	˘ 15
'02x	16	17	18	19	20	21	22	23
'03x	← 24	→ 25	ˆ 26	ˆ 27	ˆ 28	ˆ 29	30	31
'04x	ħ 32	33	34	35	\$ 36	37	38	' 39
'05x	40	41	* 42	43	, 44	= 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	58	59	< 60	— 61	> 62	63
'10x	64	65	66	67	68	69	70	71
'11x	72	73	74	75	76	∪ 77	78	○ 79
'12x	80	81	82	83	84	85	86	Ω 87
'13x	88	89	90	91	92	93	↑ 94	↓ 95
'14x	96	97	★ 98	∅ 99	† 100	101	102	103
'15x	104	105	106	107	108	109	110	111
'16x	112	113	114	115	116	117	118	119
'17x	120	121	122	123	124	125	~ 126	= 127
'20x	128	129	130	131	132	133	134	‰ 135
'21x	• 136	°C 137	\$ 138	¢ 139	f 140	© 141	W 142	N 143
'22x	G 144	P 145	£ 146	R 147	? 148	↓ 149	d 150	TM 151
'23x	‰ 152	¶ 153	B 154	Nº 155	% 156	e 157	o 158	SM 159
'24x	{ 160	} 161	¢ 162	£ 163	¤ 164	¥ 165	166	§ 167
'25x	168	© 169	ª 170	© 171	¬ 172	® 173	® 174	175
'26x	° 176	± 177	² 178	³ 179	180	µ 181	¶ 182	· 183
'27x	※ 184	¹ 185	º 186	√ 187	¼ 188	½ 189	¾ 190	€ 191
'30x	192	193	194	195	196	197	198	199
'31x	200	201	202	203	204	205	206	207
'32x	208	209	210	211	212	213	× 214	215
'33x	216	217	218	219	220	221	222	223
'34x	224	225	226	227	228	229	230	231
'35x	232	233	234	235	236	237	238	239
'36x	240	241	242	243	244	245	÷ 246	247
'37x	248	249	250	251	252	253	254	255

字体布局 13: tcrm1000. TS1 编码方案.

附录四 L^AT_EX 的宏包

- 12many** 可以使用几种格式打印连续整数集, 如 $\{3, \dots, m\}$.
- a0poster** 用于在大幅面 (A0–A3) 纸张上打印大字 (12pt 到 107pt), 而且能正确打印数学公式, 也支持 dvips.
- aastex** 美国天文协会 (AAS) 所属期刊的稿件格式文件.
- abstract** 提供对 abstract 环境的排版控制, 特别是在分成两栏的页里打印通栏的摘要.
- abstyles** BibT_EX 的自适应族使用的格式文件.
- accfonts** 包含两个操纵字库的工具, 一个用于 PostScript 的 Type 1 字库, 一个用于 T_EX 的虚拟字库, 对已知字库按用户的需要加以修改, 尤其是创建带重音符号的字母. 用 Perl 写成.
- achemso** 美国化学会用的 L^AT_EX 与 BibT_EX 格式文件.
- acm** 一些未包含在 acmconf 和 acmtrans 中的其他 ACM (美国计算机协会) 宏包.
- acmconf** 计算机协会 (ACM) 的会议录使用的格式文件.
- acmtrans** 计算机协会 (ACM) 的 Transactions 使用的格式文件.
- acronym** 用于处理缩写词, 保证每个缩写词至少出现一次全称, 还能建立缩写词的索引.
- adfathesis** 澳大利亚国防学院 (ADFA) 论文格式.
- adobeother** Adobe 非标准字库的尺寸文件.
- adrconv** BibT_EX 地址数据库转换为地址文件.
- adrlist** 在 L^AT_EX 里使用地址簿.
- ae** 用 CM 字库模拟 EC 字库生成 PDF 文件, 因此称为 AE (Almost European) 字库.
- aeguill** 在 AE 字库里添加一些引号.
- aguplus** 美国地球物理学联盟 (AGU) 的格式文件.
- aiaa** 美国航空航天学院 (AIAA) 所属期刊的稿件格式文件.
- akletter** 扩充了 L^AT_EX 的书信类 (letter), 使你能建立自己的信头, 为有窗口的信封标记折叠点等, 支持多种选项.
- alatex** 抽象 L^AT_EX, 它提供了 L^AT_EX 的所有功能, 但是可在不改动源文件的情况下改编输出的结果.
- alg** 为用 L^AT_EX 2_ε 打印计算机源程序提供了两个环境, 可以自动生成行编号, 自动缩进等.
- algorithm2e** 浮动的计算机源程序设计环境, 可以自定义关键词.
- algorithmicx** 帮助你在论文里插入计算机源程序或伪代码.
- algorithms** 为打印计算机源程序设计的环境.
- allrunes** 为打印欧洲古代诗歌设计的 L^AT_EX 包及字库.

- alnumsec** 使你可以用字母作为章节的编号, 形式多样.
- altfont** 在新字体选择方案 (NFSS) 中有 3 个族: 罗马字体、无衬线字体与打字机字体, 这个宏包使你方便地选取这几族字体使用的具体字库.
- amsfonts** 美国数学会增加的 T_EX 字库, 包括: 额外的数学符号、双线黑体 (黑板黑体)、德文花体和俄文字母等.
- amslatex** 美国数学会提供的加强 L^AT_EX 的数学公式排版能力的一些宏包, 包括 **amsmath**, **amscls** 等.
- amsrefs** 为参考文献建立类似于 BibT_EX 数据文件的数据格式, 可供 L^AT_EX 直接处理.
- amstex** 是生成 A_MS-T_EX 的宏包, 并非用于 L^AT_EX.
- answers** 生成问题 (或练习) 与解答的格式文件.
- antiqua** URW Antiqua 的 PS 字体.
- antp** PostScript Type 1 格式的 Antykwa Półtawskiego 字库, 用于波兰文.
- antt** antyktor 的更新版.
- antyktor** PostScript Type 1 格式的 Antykwa Toruńska 字库, 用于波兰文.
- anysize** 方便设定纸张与页边留空的大小.
- apa** 美国心理协会 (APA) 的格式文件.
- apacite** 与 APA 配合的 BibT_EX 格式.
- apalike** BibT_EX 的格式, 也提供 L^AT_EX 调用的宏包, 生成类似 APA 格式的文獻引用.
- apl** 印刷 APL 程序用的字库.
- appendix** 加强附录功能的宏包, 可以在任何章节之后插入附录.
- ar** 航空领域用 AR 连写表示“纵横比”, 这里提供了生成这个符号的 MetaFont 文件.
- archaic** 一些古文字的字库, 如: cypriot, etruscan, greek4cbc, greek6cbc, linearb, phoenician, runic 等.
- arcs** 在文字上下方添加圆弧线.
- arev** Arev Sans 的字体及引用它们的宏包, 此字体适宜于演示之用.
- armenian** 亚美尼亚文的宏包.
- arrayjob** 模仿程序设计语言 Fortran, Ada 或 C, 建立 array 数据结构, 并进行各种数组操作, 可用于信件管理或图形程序设计.
- arydshln** 在 array 和 tabular 环境里画水平或竖直的虚线.
- asaetr** 美国农业工程师协会会报的格式.
- ascelike** 美国土木工程师协会的文件及参考文献格式.
- ascii** 支持 IBM 的扩展 ASCII 字库.
- assignment** 用于给学生布置家庭作业或实验.
- astro** 天文 (行星) 符号.
- attachfile** 把任意文件贴到 PDF 文件里.
- augie** 提供 Augie 字体的 Type 1 字库, 是模拟非正式手写体.
- auncial-new** Artificial Uncial (一种古手写体) 的字体及引用它们的宏包.
- aurical** 是 Auriocus Kalligraphicus 的 PS 字体.
- aurora** 是 PostScript 程序, 利用它, 能分离各种彩色成分, 分别打印.
- authorindex** 生成文件中引用过的作者名及其出现页码的表.
- autotab** 从数据文件自动生成表格.
- blencoding** Bookhands 字体的 L^AT_EX 编码工具.

babel L^AT_EX 的多种语言支持包。
babelbib 利用 babel 生成多种语言的文件目录。
backgammon 打印西洋双陆棋 (backgammon) 的棋盘。
bangtex 孟加拉语 T_EX。
barcode2 生成条形码。
barcodes 条形码的字库。
bardiag 利用 Pstricks 生成条形图。
barr 生成各种交换图的宏包。
base 即 ltxbase。
bbding 包含许多 Zapf dingbats 字体的符号字库。
bbm 变体黑板黑体。
bbold 无衬线黑板黑体。
bcr Bistream Courier 的 PS 字体, 可用于替代 Adobe Courier。
beamer 用于生成供投影演示用的 pdf 文件的宏包, 功能强大。
beebe Nelson Beebe 收集的与 T_EX 有关的参考文献以及 BibT_EX 格式文件。
begriff 用于打印 Frege 的 Begriffsschrif 的包。
belleek Belleek 字库, mathtime 的替代。
bera Bera 的 PS 字体。
betababel 用于打印古希腊文。
beton 使用 Don Knuth 设计的 Concrete 字体打印文件。
bez123 为绘制线性、三次与有理二次 Bezier 曲线提供额外方便的宏包。
bezos 用于数学重音符号、张量指标等的工具。
bgreek 用于古典希腊文的排版。
bgteubner Teubner 出版社排印书籍的文件类。
bib2xhtml 把 BibT_EX 的文献目录转化成 XHTML。
bibarts 帮助制作参考文献录的宏包。
bib-fr 经典 BibT_EX 格式的法语翻译。
bibhtml 包含 Perl 脚本以及 BibT_EX 格式文件, 用于为 html 文件生成参考文献录。
使得从正文可以直接链接到有关文献。
bibleref 提供圣经内容引用的标准格式。
biblist Joachim Schrod 设计的 BibT_EX 格式。
bibtopic 把几个按主题区分的参考文献录包含到一个文件里。
bibunits 对不同章节分别生成参考文献录的格式, 此宏包与 koma-script 及 frenchb 兼容。
bigfoot 用于解决与脚注有关的问题, 例如脚注的嵌套等。
binhex 把数字转换成 2ⁿ 进制数。
biocon 打印生物物种名。
bitfield 绘制二进数位含义图。
bizcard 名片排版。
blackletter1 使 Haralambous yfrak, yswab 与 ygoth 字库为 L^AT_EX 所用。
blindtext 生成一段无意义的文本供观察新设计的格式之用。
bluesky Type 1 格式的计算机现代字体。
boites 定义一个环境, 使得加框的盒子也能换页。
bookhands book-hand 字库 (古体拉丁书写法)。
booklet 打印小册子。

booktabs 使表格更美观.
bophook 提供 `AtBeginPage`, 在每页之前执行一批命令.
borceux Francois Borceux 设计的画交换图的宏包.
bosisio 一组宏包: `dblfont`, `graphfig`, `mathcmd`, `mathenv`, `quotes`, `sobolev`.
bpchem 打印化学式.
braille 盲字支持.
breakurl `hyperref` 包的扩展, 支持链接的换行.
breqn 用于行间公式的自动分行.
bridge 用于打印桥牌图像.
brushscr BrushScript 的 PostScript Type 1 字库.
bsheaders 使得章的标题上下各画两条横线.
bundledoc 把为建立一个 L^AT_EX 文件所需的所有辅助文件打包生成 `.tar.gz` 文件, 便于跨系统转移或与同事交换.
burmese 对缅甸文的支持.
bytefield 注明各二进制位或字节的意义, 便于网络协议的说明.
c-pascal 打印 C 或 Pascal 程序.
calendar 用于创建日程表、课程表等.
calligra Peter Vanroose 的手写体字库.
calrsfs 更美观的书写体字母.
calxxxx 打印年历.
camel 参考文献录综合处理工具, 发展中.
captcont 支持延续多页的复杂插图或表格及其标题.
caption 对插图或表格的标题作更多的调节.
carlisle David Carlisle 设计的各种宏包.
casyl 打印加拿大土著方言 Cree/Inuktitut.
cbgreek 希腊字母的 MetaFont 源文件.
cc-pl 使用 Concrete 字体的波兰文补充文件.
ccaption 浮动成分的连续标题或图例等.
ccfonts 用 Concrete 字体打印文件所需的 L^AT_EX 宏包及字库定义文件.
cchess 打印中国象棋的棋子与棋盘.
cclicences 用于打印 Creative Commons 许可证图标.
ccmap cmap 的中文版本.
cct 支持 L^AT_EX 2_ε 的 CCT 中文支持宏包.
cd 打印光盘封面的宏包.
cdcover 打印光盘封面的宏包.
cdpbundl 功能强大的打印高质量商务信件(意大利格式)的宏包.
cellular 生成细胞状表格.
changebar 在 L^AT_EX 文件里标注修改部分.
chapterfolder 用来管理具有复杂文件目录的 L^AT_EX 源文件, 例如不同的章、节存放在不同文件夹里.
charter Charter 的 PS 字体.
chem-journal 化学期刊的 B_BT_EX 格式文件.
chemarrow 排版化学反应式.
chemcompounds 提供用多个数字引述化学化合物的各种格式.
chemcono 支持化学文件里的复合数字.

chemsym 打印化学符号的宏包.
cheq Adobe 国际象棋字库.
cherokee Cherokee 手写体字库.
chess 国际象棋棋盘字库.
chessfss 用于选取国际象棋的字体.
chicago 一种参考文献录的格式.
china2e 生成中国旧历符号的宏包.
chroma 列出生成各种色彩的数据.
circ 画电路图.
cirth Cirth 字库.
cite 支持对数字引用的压缩与分类.
citeref 支持参考文献录的逆向引用.
cjhebrew 支持希伯来文的宏包.
cjk 支持中、日、韩文的宏包.
cjk-fonts 提供 CJK 打印的一些字库.
cjw 打印 CJK 用的字库与宏.
clatex 支持 L^AT_EX2.09 的 CCT 宏包.
clefval 使得单词与其发音结合成一对, 输入其中之一, 另一个即同时显示.
clock 图形或文字显示的时钟.
clrscode 打印出与 CLRS 合著的“Introduction of Algorithms”书中同样格式的伪代码.
cm 计算机现代 (cm) 字库.
cm-super 计算机现代字体的 PostScript Type 1 字库.
cmmap 使得由 PDFLaTeX 生成的 PDF 文件能被搜索与复制.
cmastro 提供天文与星座符号的字库.
cmbright CM Bright 字库 (配合 CM 的无衬线字库) 的支持文件.
cmcyr 俄文字母的计算机现代字体.
cmcyralt 另一种俄文编码的支持.
cmdtrack 检查导言里的命令是否在文件里被使用过.
cmextra 美国数学会提供的计算机现代字体的附加字库.
cmpica 计算机现代 Pica 的变体.
cmsd 提供 CM 无衬线黑体的另一种界面.
codepage 支持各种代码页.
colorsep 支持用 dvips 作彩色分离.
colortab 对 tabular 与 array 环境中的单元格加彩色或阴影.
colortbl 使得在表格环境里可以使用彩色与阴影.
combine 把多个独立的文件捆扎成一个文件, 例如会议录.
commath 定义了数学公式中的一些符号以及可变大小的分隔符号, 很有用.
comment 有选择地包含或除去正文的一些部分, 可分别控制.
compactbib 生成紧凑的参考文献录.
complexity 提供打印计算复杂度符号的字体.
comprehensive T_EX 符号大全.
computational-complexity 计算复杂度杂志 (Computational Complexity) 的投稿格式文件.
concmath Concrete 数学字体的宏与字体定义文件.

concprog 排印(古典)音乐会海报.

concrete Donald E. Knuth 设计的 Concrete 罗马字库.

context ConTeXt 宏包.

context-aquamints ConTeXt 中对 AquaMints 字体的支持.

context-doc ConTeXt 说明文件.

context-lilypond ConTeXt 中的 Lilypond 编码.

contour 打印彩色轮廓.

cooking 打印菜谱.

coordsys 打印坐标系或网格.

corridx 自动校正 L^AT_EX 的索引条目的预处理程序.

couriers 把打字机字体设置为 courier 的包.

courseoutline 用于写课程讲义的包.

coursepaper 供学生写论文的包.

covington 方便排版语言学文章.

croatian 克罗地亚语字库.

crop 排印页边的切割记号.

crossreference 技术文献的交叉引用.

crossword 排印填字游戏.

crosswrđ Brian Hamilton Kelly 的填字游戏包.

crsswrđ 即 crossword.

cryst 晶体学内对称的字库.

csfonts 适合捷克斯洛伐克的计算机现代字库.

cslatex 捷克斯洛伐克用的 L^AT_EX.

csplain 捷克斯洛伐克用的 PlainT_EX.

cspfonts 捷克斯洛伐克用 PostScript 字库.

csquotes 用于排印引文的宏包, 提供许多功能, 如引文嵌套等.

csvtools 从用逗号分隔的数据文件中提取数据作为 T_EX 命令的参数.

ctable 便于排版居中的有标题或脚注的表格.

ctex ctex.org 开发的中文宏包系列.

ctib 藏文 T_EX.

cuisine 用于排印菜谱.

currvita 排印简历的宏包.

cursor 在数学环境里创建一个 L 形‘鼠标’.

curve 排印简历的宏包.

curves 在 L^AT_EX 的 picture 环境里用抛物线段画出曲线.

custom-bib 生成用户定制的 BibT_EX 参考文献格式.

cv 排印简历的宏包.

cvsty 排印简历的宏包.

cyrillic 支持使用俄文.

dancers 福尔摩斯‘跳舞的人’的字库.

dashbox 生成虚线的、分层的盒子.

dashrule 画水平虚线.

datenumber 计算天数.

datetime 设定各种打印 \today 以及当前时间的格式.

dcpic 画交换图的宏包.

de-tex-faq 德语 T_EX 的问题与解答.
decimal 按英国习惯, 把小数点打印在数字的一半高度上.
deleq 提供更有弹性的公式计数法.
devanagari 梵文 T_EX.
diagnose 用于诊断需用到的某个格式文件或 token 是否存在.
diagxy 画交换图的宏包.
dialogl 构造交互式 L^AT_EX 脚本的宏包.
dice 骰子的 2 维或 3 维图形的字库.
dichokey 构造分叉区分链 (特别用于生物学区分物种).
dictsym 提供词典里用到的符号.
din1505 参考文献录的德国标准 DIN 1505 格式.
dinat 参考文献录的德文格式.
dinbrief 德文信件의 DIN 格式.
dingbat 提供 Dingbat 字库.
directory 用 L^AT_EX 及 B_IB_TE_X 建立、维护、及使用地址簿.
dk-bib B_IB_TE_X 的挪威语格式文件.
dnaseq 打印 DNA 序列.
docmfp 使得非 L^AT_EX 的源程序, 例如 MetaFont 或 MetaPost 等也能生成说明文件.
doipubmed 增加一些用于生成参考文献录的有用命令.
dotseqn 使公式排在左边, 右端放编号, 中间是点线.
dottex 用于把嵌入源文件中的 dot 或 neato 图形文件自动处理成 PostScript 图形.
dpfloat 浮动元素按单双页放置.
draftcopy 在草稿 (draft) 输出时加上 “DRAFT” 的标记.
dramatist 用于排印刷本、诗歌等.
dratex T_EX 里使用的绘图宏包.
dropping 使每段的第一个字母加大且下沉.
dstroke N, Z, Q, R, C 的黑板黑体 (加双线).
dtk 德语 T_EX 用户间交流用的文件格式.
duerer 计算机 Duerer 字库.
dviincl 把 DVI 文件嵌入到 MetaPost 图形中.
dvipsdoc dvips 的文件.
ean 生成 EAN 条形码的字库.
easy 一族 “容易” 使用的宏.
ebezier 用一次与三次 Bernstein 多项式画曲线.
ebsthesis 欧洲商业学校的格式文件.
ec 欧洲的计算机现代字库.
ecards 用于生成屏幕智力测验题的 PDF 文件.
ecc 欧洲的 Concrete 字库.
ecltree 画树形图的宏包.
eco EC 字库的尺寸文件及虚拟字库文件.
edmac 打印供学术批评的版本.
ednotes 排印供评论的文稿的格式, 含有行号以及大量脚注.
eemeir 可以根据对象的性别切换不同的用词 (当然仅用于西文).
eepic L^AT_EX 的画图工具和 epic 的扩充. 画得更快, 内存消耗更少, 还能画椭圆、弧、样条曲线、实心圆和椭圆等.

egameps 画比赛的对阵图.

egplot 把 gnuplot 的命令封装在 L^AT_EX 源文件里, 从而能嵌入 gnuplot 生成的图形.

eiad EIAD 字库与宏包.

eijkhout 几个不相关的包, 用于: 处理数据库输出; 打印光盘标签; 生成循环等.

ellipsis 校正 T_EX 省略号的前后留空.

elmath 供具有希腊文键盘的用户使用.

elpres 用于生成电子演示材料的简单文件类.

elsevier Elsevier 科技期刊的格式文件.

elvish Tolkien Elvish 手写体字库.

emp 把 MetaPost 作图源程序封装在 L^AT_EX 里.

emulateapj 产生与天体物理杂志版面类似的预印本的格式文件.

endfloat 把所有插图集中在最后, 同时在正文应该放图的地方加上标注.

endheads 用于生成形如 “Notes to pp.xx-yy” 的页眉, 其中 xx, yy 是页码.

engpron 用 D. Jones 的符号排印英语发音.

engrec 用希腊字母作为列表的标记.

engwar Michael Urban 设计的 Tolkien Engwar 手写体字库.

enumitem 给三种罗列环境提供更多的参数选择.

envbig 在信封上打印地址.

envlab 打印信封及标签, 包括条形码.

epigraph 打印章首或章末的铭文.

epiolmec 古代中美洲一种图形文字的字库.

eplain plain 格式的扩充版.

epsdice 绘制任意大小的骰子.

eqlist 建立一个 list 环境, 使得文本的左边界由最长的标签统一决定.

eqname 公式的多种编号方式.

eqnarray 更加一般的带编号的方程组.

eqparbox 可以定义一组有相同宽度的 \parbox, 它们的公共宽度就是其中最长的自然宽度.

es-tex-faq 西班牙语 T_EX 的问题与解答.

esdiff 打印导数.

esieecv 法语履历.

esint 给出新的积分符号.

eso-pic 向每一页加入图形命令 (或背景) 的宏包.

esvect 用箭头描写向量, 这个箭头不同于计算机现代字体.

etaremun 提供一个类似于 enumerate 环境的 etaremun 环境, 其中条目编号采取递减的顺序.

etruscan 提供古文字 Etruscan 的字体.

euler 在 L^AT_EX 文件的数学模式里使用美国数学会的 Euler 字体.

eulervm Euler 虚拟数学字库.

euro 以欧元为基数的各国货币转换.

euro-ce 欧元符号以及 CE 记号.

eurofont 打印欧元符号.

euroitc ITC 的欧元符号字库.

europecv 欧洲委员会推荐的履历格式.

eurosym 欧元符号.

euxm 类似于 EUSM, 不过多了两个 Concrete Math 需要的字符.

evautofl 打印装订线或打孔的圆圈.

evweek 用于排印周历.

exam 打印试卷的宏包.

examdesign 打印试卷的宏包.

examplep 使排版输入的原文与打印输出左右并列, 而且只需输入一次.

exercise 用于在文件中插入练习及答案, 而且可使答案显示、不显示或延后出现.

expdlist L^AT_EX 的 description 的功能扩充.

expl3 尚在试验中的 L^AT_EX3 项目.

export 输入和输出 L^AT_EX 寄存器的值.

expressg MetaPost 的宏包, 适用于画含有方框、直线和注释的图.

extrarrows 使数学模式中箭头的长度能任意伸展.

extract 把源文件的一些命令或环境输出到一个目标文件.

extsizes 扩充 article 与 report 类, 使它们的基础字体尺寸可以是 8–20 pt.

facsimile 用于创建传真文稿.

fancybox 扩展盒子的样式和功能.

fancyhdr 扩展 L^AT_EX 里对页头和页脚的控制.

fancynum 用于打印数字, 特别是计算机提供的数字的宏包.

fancyref 扩充交叉引用的功能.

fancyvrb 对抄录环境的加强.

fax 用于传真的文件类.

fbthesis 德国 Dortmund 大学计算机系的学位论文格式.

fc 非洲语言的字库.

feyn 用于绘制放在公式里面的、相对简单的 Feynman 图.

feynmf 创建较大、较复杂的 Feynman 图.

figbib 在 BibT_EX 数据库里管理图形.

figsize 会自动确定 (几个) 插图的大小, 使它们充满一页或一个指定大小的范围.

filecontents L^AT_EX 2_ε 的 filecontents 与 filecontents* 环境的改进版.

finbib BibT_EX 的芬兰语格式文件.

fink L^AT_EX 2_ε 的文件名管理器.

firststeps 一些例子.

fixfoot 使同一脚注内容多次使用.

fixme 用于 draft 模式, 在需补充修改的地方插入 fixme 标记.

flabels 打印更美观的标签.

flacards 打印快速问答卡 (正面是问题, 反面是答案).

flagderiv 用于生成逻辑推导的直观格式.

flashcards 打印快速问答卡 (正面是问题, 反面是答案).

float 改进浮动单元 (插图或表格) 的界面.

floatflt 使正文在浮动单元的旁边绕行.

floatrow 对同类浮动对象建立一致的样式.

flowfram 生成以文本为内容的框图及流程图, 特别适用于张贴材料, 小册子, 杂志等.

fltpage 定义新的环境, 使得插图或表格的标题放在前或后一页.

fltpoint 使 T_EX 能做浮点算术运算.

fmp Functional MetaPost (FMP) 是 MetaPost 的高级界面, 使得用户能用 Haskell 语言编写绘图程序. 本宏包是把 FMP 源程序嵌入 L^AT_EX, 通过两轮处理后就能自动把图像嵌入文件.

fnbreak 检测被分割在不同页内的脚注.

fncychap 提供了 6 种预定义的章的标题格式.

foillhtml 提供了 FoilTeX 与 LaTeX2HTML 的集成.

foiltex FoilTeX.

fontinst 把 Adobe 的字库尺寸文件转换成 T_EX 的尺寸文件以及虚拟字库文件.

fontinstallationguide 告诉你如何安装新的 PS 字库.

fontsamplers TeX Live 光盘所提供的字样表.

fonttable 用于在 L^AT_EX 文件内打印字符表.

fonttools 为 L^AT_EX 中使用 ttf 字体提供方便的一些工具.

footbib 把参考文件作为脚注的宏包.

footmisc 收集了各种关于脚注的宏包的功能作为本宏包的选项.

footnpag 允许每一页的脚注从 1 开始编号.

formula 支持物理符号, 保证它们在文本与数学模式有同样的形状, 包含一些预定义的物理单位.

formular 用于打印需要手填的表格.

fourier 提供与 Adobe Utopia 配套的字库.

fouriernc 为新世纪小学课本提供配套的 Fourier 数学字体.

fp 提供高精度定点实数的一些算术运算.

fpl URW Palladio L 的 SC 与 OsF 字库.

frankenstein 一族有各种功能的 L^AT_EX 的宏包. 如: abbrevs, achicago, achicago-bibstyle, attrib, bits, blkcntrl, compsci, dialogue, drama, includex, lips, more-defs, newclude, slemph, titles 等.

frcursive 法文手写体字库.

frenchle 供排印法文文章用.

fribrief 用于写信的 L^AT_EX 类.

fullbck 与 letter 类联用, 使得信件的内容都向左端对齐.

fullpict 用于画整页的图.

functan 用于在泛函分析和偏微分方程里处理函数空间.

fundus 使 L^AT_EX 能使用一些字库族.

futhark 一种古代北欧语的字库.

g-brief 信件的格式.

galley L^AT_EX3 的宏包.

galois 用二维格式写 Galois 联络.

gatech-thesis 乔治亚技术学院的学位论文格式.

gauss 打印矩阵运算.

gb4e 政府装订格式.

gchords 排印吉他的弦图.

genealogy 宗谱学需用符号的字库.

genmisc 各种格式的小文件.

genmpage 扩展 minipage 环境.

geometry 很容易设定输出文件的各种版面尺寸.

geomsty 包含多种功能的宏包, 如嵌入 PostScript 插图, 自动创建索引条目以及交叉引用, 易于定义类似于定理的环境等.

german 支持德文排版.

germbib BibT_EX 标准格式的德文版.

germdoc 德语的文件.

ginpenc 用于输入德语变音字母.

glonti LCY 和 T2A 编码的虚拟字库, 由 CM 和 CMCYR 字库组成.

gloss 利用 BibT_EX 创建术语表.

glossary 方便生成词汇表或术语表.

gn-logic 便于某些类型逻辑公式的排版.

go 打印围棋.

gost 按 GOST 标准生成 BibT_EX 的格式 (俄文用).

gothic Yannis Haralambous 的哥德体字母.

gothict1 yfrak, yswab 和 ygoth 的 PostScript Type 1 字库.

gradback 渐变的背景色彩.

graphics 图形包, 书中已有介绍.

graphicx-psmin 标准绘图包的补充, 使得重复使用的 ps, eps 图形只需输入一次.

greek 希腊文输入的编码文件.

greek4cbc 公元前 4 世纪的一种希腊字体.

greek6cbc 公元前 6 世纪的一种希腊字体.

greek4doc 即 greekdoc.

greekdoc 希腊文的 T_EX 导引.

greenpoint The Green Point 字体的 MetaFont 文件.

grfpaste 使能插入 dvi 的片断, 不过需要用到 dvipaste 程序.

grnumalt 提供古雅典的计数系统, 可用来打印计数器里的数字.

grotesq URW Grotesk Bold 的 PS 字库.

grtimes 可用 Times New Roman 的希腊字母打印希腊文, 不过不附字库.

grverb 希腊文的 verbatim 环境.

guides T_EX, MetaFont, MetaPost 等的指导书.

guit 用于打印意大利 T_EX 用户协会徽标等.

guitar 用于打印吉他曲谱.

gustlib 供波兰文使用的宏包.

ha-prosper prosper 宏包的补丁, 用于生成供投影仪演示的 PDF 文件.

hands Hands 字库.

hanging 用于排版悬挂式段落, 即从第二行开始缩进的段落.

har2nat 使原来使用 harvard 包的参考文献格式改成 natbib 包的格式.

harpoon 提供更多的 harpoon (→) 符号.

harvard Harvard 参考文献格式.

harvmac Paul Ginsparg 的写科学论文的 Harvard 宏包.

hatching MetaPost 的宏包, 用于在闭路径的内部加影线.

hc 以 KOMA-Script 和 seminar 类为基础的文件类, 用于代替 L^AT_EX 原有的文件类.

hebclassc 用于希伯来语的宏包.

hep 用于打印高能物理论文的宏包.

hepnames 提供高能粒子的名称.

hepparticles 提供高能粒子的名称.

hepthesis 用于打印硕士博士学位论文的文件类。

hepunits 提供高能物理中用到的单位。

hfbright hf Bright 的 PS 字库。

hfoldsty 供使用欧洲计算机现代字体 (其中数字是古体的) 的配套文件。

hh Herman Haverkort 创建的一族宏包, 有多种功能。

hhtensor 提供各种格式的向量、矩阵、张量的命令。

hieroglf 提供古埃及象形文字的 MetaFont 源文件。

hilowres 能简化高分辨率图形的低分辨率版本的嵌入。

histogr 用 L^AT_EX 的图形环境画柱形图。

hitec 供高科技公司的技术文件使用的类。

hoekwater 由 Taco Hoekwater 转换成的 PostScript 字库, 包括 logo, manfnt, rsfs, stmaryrd, wasy, wasy2, xipa 等。

hpsdiss 瑞士联邦技术学院的论文格式。

humanist Humanist 古字体。

hvfloat 旋转浮动对象及其标题。

hvmath 对使用 Micropress HV-Math 字库的支持。

hyper 支持超文本交叉引用。

hyperref 支持超文本交叉引用。

hyphenat 开启或关闭添加连字符。

ibm 虚拟字库。

ibycus-babel 供使用古希腊文的宏包。

icmsec-thesis 张林波开发的中科院计算所博硕士学位论文宏包。

ieeeconf IEEE 计算机出版社的会议录用的格式文件。

ieeepes IEEE 动力工程学会学报的格式文件。

ieeetran IEEE 学报、会议录的 L^AT_EX 类。

ifmslide 通过 PDFLaTeX 及 L^AT_EX 做演示, 书中有介绍。

ifsym 登山、电子、气象、几何等使用的符号。

igo 围棋包 go 的改进。

IMA 英国应用数学学会的 L^AT_EX 宏包。

imac 国际模式分析大会格式。

index 改进 L^AT_EX 原有的索引功能。

inlinebib 内嵌的 \cite。

insbox 把图形或盒子插入段落的 T_EX 宏包。

invoice 写发票的环境。

iop 英国物理学会的 L^AT_EX 宏包。

ipa 用于打印国际音标。

iso 用于按国际标准 ISO 排版。

iso10303 用于按国际标准 ISO 10303 排版。

isodate 对 \today 的输出作调整。

isorot 用于旋转文件的各种元素。

isotope 用于同位素的排版。

italian-doc 用意大利文写的 T_EX 教材。

iwona 一种无衬线字体。

jadetex 用于实现 Jade DSSSL 输出的打印。

jasthesis 英国 Bristol 大学的论文格式。

jhep 用于 JHEP 格式原稿的排版。
jknapen Joerg Knappen 创建的各种宏包, 主要用于各种额外的字体。
jkthesis Joerg Knappen 创建的学位论文格式。
jsmisc Joachim Schrod 创建的各种有用的宏包。
jura 用于打印德文法律文件。
juraabbrev 用于打印德文法律文件。
jurabib 与 jura 配合的 BibT_EX 格式。
juramisc 用于打印德文法律文书。
jurarsp 用于打印德文法律文书。
kalender 用于创建德文日历。
karnaugh 用于打印 Karnaugh-Veitch 逻辑图。
kdgreek 希腊文 T_EX。
kerkis 一种希腊文字库。
kerntest 用于微调某种字体的字母间隙。
keystroke 用图形表示键盘的各键。
kixfont KIX 码。
kluwer Kluwer 期刊的格式文件。
knuth Knuth 本人的文件, 包括 T_EXbook 和 MetaFont book。
koma-script 这是一族宏包, 可以代替 article/report/book 类, 功能更灵活, 也更考虑到排印。
ktv-texdata 便于给学生布置作业。
kurier 一种无衬线字体。
kuvio 绘图的宏以及打印交换图的字库。
l2tabu 介绍一些过时的宏包与命令 (德文版)。
l2tabu-english 介绍一些过时的宏包与命令 (英文版)。
l2tabu-french 介绍一些过时的宏包与命令 (法文版)。
l2tabu-italian 介绍一些过时的宏包与命令 (意大利文版)。
labbook 用于排印实验室日志。
labels 用于打印粘贴的标签或名片。
lamstex A_MS-T_EX 与 L^AT_EX 的结合。
lastpage 用于需要引用最后一页的页码的情形, 例如共几页中的第几页。
latex2e-help-texinfo L^AT_EX 的参考文献。
latex2html 把 L^AT_EX 文件转换成 HTML 文件。
latex4wp 向字处理软件使用者介绍 L^AT_EX。
latexdiff 比较两个 L^AT_EX 源文件, 比较它们的实质性差别。
latexfonts L^AT_EX 的字库。
latexmk 用 Perl 写的工具软件, 自动多次执行 L^AT_EX 以保证交叉引用等, 也支持打印与预览。
latexmp MetaPost 的包, 用以提高基于 L^AT_EX 的打印能力。
layaureo 更好利用 A4 纸的版面。
layouts 图解版面设置的各种要素。
lazylist 用以分析 T_EX 是如何消化其输入文本的。
lcd 模拟液晶显示屏。
lcg 生成随机整数。
le 使 L^AT_EX “法语化”的包。

leaflet 一张纸上打印多页, 供折叠装订成小册子.
leawood 使用 ITC Leawood (商业性的) 字库的每套文件.
ledmac 用于打印供学术批评的版本.
leftidx 在数学模式里把上下标打印在左边.
lettre 用于写信的文件类.
lettrine 打印下沉的大写字母.
levy 使用 Silvio Levy 的希腊字库的宏包.
lexikon 用于生成双语词典.
lgreek 使用 Silvio Levy 的希腊字库的宏包.
lh Olga Lapko 的 LH 字库 (西里尔语言).
lhcyr 包含 3 种格式, 用于俄语排版.
lhelp 包含许多有用的宏.
limap 按信息映射法的需要排印映射与块.
lineno 在选定的段落里加上行号.
linguex 方便语言学例子的排版.
lipsum 打印 Lorem Ipsum 的指定段落.
listbib 生成 B_BT_EX 格式数据库的目录.
listings 用于打印计算机源程序.
listliketab 排印类似于列表的表格.
literat 对 Literaturnaya 字库的支持.
lkort 荷兰文的 L^AT_EX 导引.
lm 拉丁现代字体的 PS 字库.
localloc 使 T_EX 的记录器局部化.
logic 用于画逻辑图的字库.
logpap 用于画坐标纸, 其中可以包含对数坐标.
longdiv 作长除法.
lshort-english 英文版 L^AT_EX 2_ε 导引, 类似还有其他语言的版本, 不一一列举.
ltablex 修改 tabularx 环境, 既保持在固定表宽的条件下自动调节列宽的功能, 又能不限制在一页里.
ltabptch Longtable 包的补丁.
ltxbase L^AT_EX 的核心.
ltxindex 用于生成索引文件.
ltxmisc 各种 L^AT_EX 的格式.
lucida 使 Lucida Bright 能为 L^AT_EX 所用.
luxi 一族打字机字体.
ly1 对 LY1 L^AT_EX 编码的支持.
mailing 用于发送大量内容相同的邮件.
makebox 改进 \makebox 命令, 使得盒子的宽度能等于指定文本的宽度.
makecirc 画电路图用的 MetaPost 库.
makecmds 提供重新定义命令或环境的命令.
makedtx 帮助 L^AT_EX 宏包开发者生成 dtx 文件.
makeglos 把词汇表或术语表加入 L^AT_EX 文件中.
makeplot 利用 pstricks 从 Matlab 输出的图形数据生成图像.
makor 希伯来语排版系统.
malayalam Malayalam 文字库.

maltese 支持马耳他文的输入.

malvern 一个新的无衬线字库族.

manfnt 在 L^AT_EX 里使用 T_EXbook 书中符号的宏包.

manjutex 支持满文.

manuscript 模拟打字机打印效果.

mapcodes 支持多种字符集与编码.

maple MAPLE 通讯的格式与例子.

marvosym Martin Vogel 的 PostScript Type 1 字库, 其中包括欧洲货币符号, 结构工程符号, 钢结构符号, 天文符号等.

mathabx 与几种正文字体 (西文) 配合的数学符号字体.

mathcomp 为文本伴侣字库 (textcomp) 提供一些数学模式的字符.

mathdesign 与几种正文字体 (西文) 配合的数学符号字体.

mathematica 使 Mathematica 3.0 随带的 PostScript 字库能被 T_EX 使用的虚拟字库.

mathpazo 配合 PostScript 的 Palatino 字库使用的 Pazo 数学符号字库.

mathtime 支持在 L^AT_EX 里使用 Mathtime 字库 (需购买).

mattens Hassenpflug 设计的矩阵张量符号.

maybemath 根据上下文选择数学字体是粗体还是斜体.

m-bib ConT_EXt 的参考文献模式.

mcaption 把插图或表格的标题放在页边.

mceinleger 用于打印录音带盒.

mcite 合并多重引用.

mdwtools Mark Wooding 创建的多个工具.

memoir 用于排版各种书籍.

mentis 德国 Mentis 出版社的格式.

menu 用于写作软件说明书时介绍如何在菜单里依次选取选项.

metaobj 基于 MetaPost 的面向对象的绘图高级语言.

metaplot 把外部的描点图像纳入 MetaPost 的宏包.

metapost-examples 用 MetaPost 绘图的例子.

metatex 是 Plain T_EX 和 MetaFont 的宏包, 在一个源文件里可以同时包含文本与图形.

method 打印编程语言 method 以及变量声明.

mex T_EX 的波兰文格式.

mf-ps 实现 MetaFont 与 PostScript 源文件的转换.

mff 提供对计算机现代字体的各种参数的控制, 使 T_EX 生成 mf 文件, 再用 MetaFont 生成字库.

mflogo 支持使用 MetaFontbook 的图标.

mfnfss 额外字库如 yinit 和 ygoth 的描述文件.

mfpic 利用 MetaFont 和 MetaPost 画图.

mfnc 使 MetaFont 的源文件打印得更便于阅读.

mh 加强数学公式的排版功能.

mhchem 用于排版化学分子式与化学公式.

mhequ 建立公式组的多列对齐环境, 并对其中公式分别编号.

mhs 数学家年表.

microtype 与 pdfT_EX 合用, 可根据需要对打印字符的宽度作微调.

midnight 一组有用的工具.
miller 打印 Miller 指数.
miniplot 帮助 L^AT_EX 用户完成 EPS 插图的排版.
minitoc 为每一章生成一个目录.
minutes 打印会议录的宏包.
misc 一些杂类宏包.
misc209 一组 L^AT_EX 2.09 时代的格式文件, 不过 L^AT_EX 2_ε 仍能使用.
mla-paper 与 pdfL^AT_EX 合用, 方便学生打印作业或论文.
mltex 对 ML^ATeX 的支持.
mnras 皇家天文学会的格式.
mnsymbol 与 Adobe Minion 配合的符号.
modroman 把数字写成小写罗马字母.
montex 支持蒙古文的 T_EX.
morehelp 加强 L^AT_EX 的出错信息.
moresize 允许使用直至 35.83pt 的字体.
moreverb 对抄录环境 (verbatim) 的加强.
morse 打印 Morse 码.
movie15 把多媒体片断包含到 PDF 文件中.
mparhack 纠正命令 \marginpar 的漏洞.
mpattern 在 MetaPost 里定义与使用图案的宏包.
mpfnmark 定义 minipage 环境里的脚注样式.
ms Martin Schröder 创建的各种 L^AT_EX 宏包.
msc 打印信息序列图的宏包.
msg 输出被使用的宏包或文件类的信息.
mslapa 模仿美国心理协会的引用格式的 L^ATeX 与 BibT_EX 格式文件.
mtbe Arvind Borde 的书 “Mathematical TeX by Example” 所用的格式文件.
mtgreek 与 MathTime 合用的直力或斜体的希腊字母.
multenum 多列的枚举格式.
multi 多重 PS 头文件.
multibbl 与 BibT_EX 合用, 重新定义了参考文献的命令.
multibib 允许在一个文件里引用几个参考文献目录里的文献.
multicap 在分栏的环境里对浮动单元的标题作格式化.
multido 在一般 T_EX 中使用的循环命令.
multirow 使得一个表格的某些单元伸展几行.
musictex 用 T_EX 打印乐谱.
musixps MusiX_TE_X 的 PostScript Type 1 字库.
musixtex 扩展的 MusicT_EX.
muthesis 曼彻斯特大学计算机系的学位论文及项目报告的格式文件.
mwcls 根据波兰文传统风格设计的文件类.
mwrite 可把信息写到任意多个文件里.
nag 发现源文件中已经过时或废弃的命令.
namespc 用于宣布局部的 T_EX 命令, 并且能在以后的段落中被重复使用.
natbib 能同时产生 “作者-年份” 或数字编号的文献引用格式.
nath 使得同一个数学公式在文章的不同位置用不同的格式打印, 例如出现在行内的分式打印成 $1/x$ 的形式, 而在单列时打印成 $\frac{1}{x}$ 的形式 (输入均为 \frac 1x).

nature Nature 杂志的格式文件.

ncclatex A. Rozhenko 创建的 L^AT_EX 类与宏包.

ncctools A. Rozhenko 创建的各种 L^AT_EX 宏包.

nddiss 美国 Notre Dame 大学打印学位论文的格式类.

newalg 用于计算机程序排版的宏包.

newfile 使用户能在 T_EX 运行时读或写新文件.

newlfn 用来写信件、传真及备忘录.

newsletr 制作时事通讯的宏包.

newthm 对定理环境的改进, 不过本宏包已被 ntheorem 取代.

newvbtm 用于定义你自己的抄录环境.

niceframe 可画出围绕文字的各种图框.

nkarta 包含地图符号的 Karta 的 MetaFont 字库.

nomenc1 利用 MakeIndex 产生符号列表.

nomentbl 利用 longtable 以及 nomencl 产生符号列表.

nonfloat 对插图与表格环境作调整, 使得它们的标题仍然居中.

notes 用图像突出一段文本.

nrc 加拿大物理杂志的格式.

ntabbing 对制表位环境的扩展, 支持对行自动编号.

ntgclass 荷兰用户协会修改过的 L^AT_EX 类文件.

ntheorem 对定理环境的加强.

nudtbook 毛紫阳开发的国防科技大学博硕士学位论文宏包.

numline 编行号的宏包.

numprint 使打印出来的数字每 3 位一组分隔开.

oands 用于翻译古手稿的字模.

oberdiek 一些有用的宏包和格式文件.

objectz 用于打印 Object Z 的宏包.

oca OCR 字库.

ocr-a OCR-A 字库.

octavo 写书用的类文件.

oesch 澳大利亚学派手写体的 MetaFont 文件.

ofs 用于管理大型字库.

ogham Ogham 古英文字库.

ogonek 支持波兰文排印.

oldstyle 载入 cmmti, cmmib 字库以产生老式数字.

onlyamsmath 在使用 amsmath.sty 时禁用 T_EX 与 L^AT_EX 的数学环境, 主要用于写类文件.

opcit 用脚注的形式列出参考文献表.

orderrefs 把文献目录按引用次序重排.

orsc 吴凌云开发的中国运筹学会论文模板.

osa BibT_EX 的一种格式文件.

osmanian 写索马里文用的字库.

ot2cyr 使用 OT2 西里尔文编码的宏包.

othello 打印 othello 棋的棋盘.

other 包含一些 AMS 的格式文件.

outline 提供一个写概要的环境.

outliner 使得改变章节的层次变得容易.

outlines 提供多达 4 层的罗列环境可用于写提纲.

overpic 在插入的外部图形的指定位置上放置 L^AT_EX 的命令.

oxford BibT_EX 的 oxford 格式文件.

pacioli Fra Luca de Pacioli 在 1497 年设计的字库, 只有大写字母及标点符号.

pageno 重新定义 'plain' 页面版式, 页码可以根据选项放在各种位置.

pagenote 把编号的脚注排印在单独的页上.

pandora Pandora 字库族.

paper 由 article 导出的类, 适用于期刊的文章, 有更多功能.

paralist 在段落内的罗列或枚举.

parallel 提供一个并列的环境, 适用于两种语言的对照.

parese 使你很容易地输入希腊字母.

parrun 在同一页上并列排印两个各自独立的文本.

passivetex 支持 XML/SGML 排版的宏包.

patch 用于包管理的宏.

patchcmd 在一个宏的头部或尾部添加一些内容.

pawpict 用于插入 PAW (Physics Analysis Workstation) 产生的图形.

pb-diagram 利用 LAMSTeX 或 Xy-pic 字库的图形包.

pbox 建立宽度可变的 `\parbox`.

pbsheet 用于打印给学生用的作业题或试题纸.

pclnfs 提供对 PCL 打印机的字体支持.

pdcmac Damian Cugley 的工具宏.

pdfcprot 对字符突出的调节.

pdfcrop 除去输入 PDF 文件的页边空白.

pdfpages 插入外部 PDF 文件的指定页.

pdfscreen 生成适合屏幕阅读的 PDF 文件, 参见本书 13.3.1 节.

pdfslide 生成适合演示的 PDF 文件, 参见本书 13.3.2 节.

pdftricks 由于 psTricks 宏包不能用于 pdfL^AT_EX, 而本宏包与 pdfL^AT_EX 联用则可提供 psTricks 的功能.

pdfwin 生成适合演示的 PDF 文件, 类似 pdfslide.

pecha 支持打印藏文古典文书的类.

perltex 用 Perl 语言定义 L^AT_EX 的宏, 使得在 T_EX 里也能使用 Perl 的功能.

permute 可以输入、输出置换, 并进行计算.

peterw Peter Wilson 建立的包.

petri-nets 用于 Petri 网络以及相关模式的宏包.

pgf L^AT_EX 的绘图包, 类似于 Pstricks.

phaistos 配合使用 Phaistos 字样的文件.

phoenician 支持腓尼基文书的字体.

phonetic 国际音标的 MetaFont.

photo 建立一个名为“照片”的新浮动类型, 类似于表格与插图.

physe PHYSE 格式, 物理论文的一种格式.

phyzzx 物理学家用的一种 T_EX 格式.

picinpar 把图形插入段落内 (Piet van Oostrum 不推荐这个包, 推荐用 picins).

picins 把图形插入段落内.

pictex T_EX 和 L^AT_EX 的绘图包.

pictex2 向标准的 PiCTeX 里加入相对坐标和绘图时用线段代替点.

piechartmp 利用 MetaPost 画圆形统计图表.

piff Mike Piff 创建的工具包.

pittetd Pittsburg 大学的电子版学位论文格式文件.

pitthesis 匹兹堡大学学位论文格式.

pkfix Perl 脚本程序, 试图用 type 1 字库替换 ps 文件中的 pk 字库.

pl 用 L^AT_EX 写带注解的 Prolog 程序.

placeins 控制浮动图表的位置.

plari 用于戏剧脚本排版的文件类.

plates 把彩色图形集中在一起打印.

platex 波兰文的 L^AT_EX.

play 用于戏剧的排版.

plcalendar 用于日历的排版.

plfonts CM 字库的波兰文扩展.

plgraph L^AT_EX 的绘图包, 也能用于 PlainT_EX.

plpatch 用于对已有的宏作微小修改.

plpsfont 波兰文字体文件.

pgraph L^AT_EX 绘图环境的扩展.

poemscol 用于打印诗歌的征求意见稿.

polish-doc L^AT_EX 文件的波兰文版.

polyglot 支持多种语言.

polynom 用于有理系数多项式的打印及运算.

polytable 建立更灵活的表格环境.

postcards 用于成批寄发明信片.

powerdot 用于生成演示材料的宏包, 功能十分强大.

ppower4 对用 pdfL^AT_EX 创建的 PDF 演示作后处理, 以提供动态效果. 软件使用了 Java.

ppr-prv 配合 Prosper 包, 打印投影片 Java.

preprint 用于打印预印本的一些格式文件.

prettyref 为 L^AT_EX 的标签-引用系统添加更多功能, 兼容 hyperref 及其他宏包.

preview 与 dvips 配合可把排版的结果提取成一个个 eps 图形.

probsoln 使教师可以选择布置给学生的习题及其答案.

prociagssymp 国际测地学协会会议录格式.

progkeys 包含两个格式文件, 分别用于程序的排版以及关键词的突出打印.

program 用于打印程序及算法.

progress 在编译文件时生成一个 html 文件显示文件各部分的进展状态.

proofs 建立证明树的宏.

prosper 打印高质量投影片的 L^AT_EX 类.

protex 照原样打印程序的包.

protocol 打印会议议程.

protosem proto-Semitic 古文字的字体的.

ps4pdf 在 pdfL^AT_EX 文件里使用 PostScript 命令.

psafm 标准 PostScript 字库的 afm 文件.

pseudocode 用更自然的方式描述一个算法.

psfig 用于插入 eps 图形的包.

psfrag 允许把 L^AT_EX 的各种内容复叠于 eps 图形之上。
psfragx psfrag 包的扩展。
psgo 用 PostScript 画围棋盘。
psizzl SLAC 的 T_EX 格式。
pslatex 以 PostScript 字库作为默认字体的 L^AT_EX。
psnfss 对通用 PostScript 字库的支持文件。
psnfssx PostScript 字库的附加的格式与编码。
pspicture 把 L^AT_EX 的 picture 环境的图形用 PostScript 命令实现, 这样就取消了原来 picture 环境的许多限制。
pst-3d 建立 3D 效果的 PSTricks 包。
pst-3dplot 利用 Pstriks 画 3 维曲线或数学图形。
pst-bar 利用 Pstriks 画条形图。
pst-barcode 利用 PostScript 的计算功能打印条形码。
pst-blur 利用 Pstriks 画阴影。
pst-calendar 利用 PSTricks 创建有想像力的日历。
pst-circ 利用 Pstriks 画电路图。
pst-eucl 利用 PSTricks 画欧氏几何的图形。
pst-fr3d 利用 Pstriks 画 3 维长方体, 尤其适合画立体按键。
pst-func 利用 Pstriks 画函数图形。
pst-ghsb 利用 Pstriks 画渐变的色彩。
pst-gr3d 利用 Pstriks 画 3 维网格。
pst-infixplot 利用 Pstriks 画函数图像。
pst-jftree 利用 Pstriks 画树形图。
pst-labo 利用 PSTricks 画化学实验室物件。
pst-lens 利用 Pstriks 画透镜。
pst-light3d 利用 Pstriks 画 3 维光线效果。
pst-math 在 Pstriks 里加入数学算子如初等函数等。
pst-optic 利用 Pstriks 画光路图, 有透镜、镜子等。
pst-osci 利用 Pstriks 画振动图。
pst-pdf 使 PSTricks 生成的图形能用于 pdf 文件。
pst-poly 利用 Pstriks 画多边形。
pst-slpe 利用 Pstriks 画渐变的色彩。
pst-uml 利用 Pstriks 画 UML (Unified Modelling Language) 类型的图。
pst-vue3d 一个支持 3D 视图的 PSTricks 宏包。
pstricks 使用 PostScript 的各种宏, 能提供许多绘图功能。
pstricks-add Pstriks 的追加文件与补丁。
ptptex Progress of Theoretical Physics 杂志的格式。
punk Donald Knuth 的 punk 字库。
pxfonts Adobe Palatino (或 URWPalladioL) 的 T_EX 符号的 Type 1 字库, 可用于 OT1, T1, TS1, LY1 编码。
qcm 用于生成多重选择题。
qfonts 波兰文用的一组 PostScript 字库。
qobitree 打印‘树’使用的宏。
qpx 与 qfonts 联用的数学字体文件。
qsymbols 借助单引号与双引号定义各种数学符号助记符。

qtx 与 **qfonts** 联用的数学字体文件.

quotchap 在章的标题页用引文作装饰.

qxcn 适合波兰文用的扩展 CM 字库.

r-und-s 把标示化学品安全警告的“R-S 规则”打印成文字.

rccol 使表格里的数字居中向右对齐.

rct 在 L^AT_EX 文件里使用版本控制系统的标记.

rctinfo 对版本控制系统信息的支持.

realcalc 执行实数算术运算的宏.

rectopma 使 \title 与 \author 的内容能在文件各处重复使用.

refcheck 检查标签与引用.

references 对支持 L^AT_EX/BibT_EX 的书目管理软件 REFERENCES 的安装介绍.

refman 用于写技术参考手册的格式.

refstyle 支持多种引用格式.

regcount 显示 T_EX 的寄存器的状态.

register 用于打印寄存器各数位的意义的宏包.

relen 提供一个较宽松的字体编码环境, 使字库设计者易于插入连字或重音符.

repeatindex 当索引表换页或换栏时, 重复跨页的项目.

resume 打印简历的格式.

revtex4 美国物理协会的格式文件.

rleptf 与 **epstex** 合用的宏包, 使得 eps 文件里的 PostScript 标签可被替换成 T_EX 标签.

rmpage 帮助用户改变版面样式.

robustindex 使得 \pageref 里的页码保持正确.

romannum 用罗马数字代替阿拉伯数字.

rotating 建立在 L^AT_EX 标准图形包的基础上, 能进行各种形式的旋转, 包括整个插图、表格、标题的旋转.

rotfloat 结合 float 与 rotation 包的功能, 使得各种浮动对象都能旋转.

rotpages 把某些页旋转 180°, 并打乱页次.

rplain 重新定义 ‘plain’ 页面版式, 现在页码可以出现在右下角.

rrgtrees 生成语言学的树形图.

rsfs 提供一种手写体大写字母的字库.

rtkinenc 设置“激活”字符, 使得不能打印的 ASCII 字符被转换成相应的 L^AT_EX 命令.

ruhyphen 俄文换行规则.

runic 一种盎格鲁撒克逊古文字字体.

sae SAE 的论文格式.

sanskrit 用于打印梵文.

sauerj J. Sauer 开发的一些工具包.

sauter CM 字库的扩充, 并提供参数的选择方法.

sauterfonts 提供使用 Sauter (或 Knappen, Holin) 字库的字体定义文件.

savefnmark 使同一个脚注标记重复使用.

savesym 同一个符号命令被不同的包定义成不同的图形, 现在要使这些不同的图形能同时被使用.

savetrees 使得一页里能容纳更多的内容.

scale 用 sqrt2 或 magstep2 放大文件.

scalebar 在地图、图表或照片里创建缩放尺。
schedule 利用图形环境自动格式化每周日程表。
scientificpaper 用了这个包后,正文被排成两栏,摘要通栏。
sciposter 排印 A3 纸的招贴。
sciwordconv 建立 Scientific Word/Scientific WorkPlace 与 L^AT_EX 的衔接。
script 各种报告/书籍格式。
sectionbox 在章节标题外面加各种彩色边框,特别适用于打印张贴用的学术报告。
sectsty 改变章节标题的样式。
semantic 便于打印语义学和编译器的符号。
semaphor 旗语字母的 MetaFont 文件。
seminar 生成演示用的透明片。
semioneside 把正文的内容打印在右边页面上,只让特选的内容打印在左边页面上。
setspace 设定行距,例如两倍或一倍半行距。
sf298 生成递交报告用的标准格式 298。
sffms 用于写(英文)科幻小说的原稿。
sgame 画对策游戏。
shadbox 为盒子的背景打阴影。
shadethm 给定理打阴影背景。
shalom 供希伯来文使用。
shapepar 排印指定样式的段落。
shorttoc 具有不同深度的目录。
showdim 提供命令用于打印 T_EX 的长度值。
showframe 画出文本或页边空白等的边框,为检验版面之用。
showlabels 把标签的名字打印在草稿的页边。
siam SIAM 出版物的格式。
sidecap 把标题打印在插图或表格的旁边。
sides 用于打印剧本的文件类。
siggraph 美国计算机协会计算机绘图专业组年会的论文格式。
simpsons Simpsons 字符的 MetaFont 文件。
sistyle 按 ISO 标准打印 SI 单位。
siunits 按国际单位系统的标准打印物理单位。
skak 打印国际象棋棋盘。
skaknew 用新字库替代标准的 skak 字库。
skull 打印骷髅头。
slantsc 打印斜体的小型大写字母。
slashbox 在表格里画斜线。
slidenotes 打印带有注释的投影片。
slideshow 利用 MetaPost 和 GhostScript 制作 PDF 演示资料。
smallcap 支持使用小型大写字母。
smalltableof 用于处理不属于某个章节的图表。
smartref 扩展 L^AT_EX 的 \ref 功能。
smflatex 法国数学会的格式文件。
snapshot 列举一个 L^AT_EX 文件依赖的外部文件。
songbook 打印歌词的包。

soul 使每个字母在一定的宽度内均匀散布, 支持对单词加上下划线及换行.

spain 西班牙文的参考文献格式.

sparklines 引入 sparkline 环境, 使得可使用启发性图案.

spie SPIE 会议录的格式文件.

splitbib 允许参考文献录被分成几个部分.

springer Springer 的科技论文、图书排版宏包.

sprite 利用 T_EX 作点阵图.

sqr caps 一种古典的英文字体.

srcltx 在 DVI 文件里插入信息, 使得能从 DVI 浏览器作逆向搜索 (跳回 T_EX 源文件) 或正向搜索.

sseq 用于打印谱序列.

ssquote 帮助用户使用 ‘cmssq’ (计算现代无衬线引文字体) 的包.

stack 建立一个独立于 T_EX 的堆栈管理系统, 仅用于宏包开发.

stage 用于写戏剧剧本.

startex 帮助学生写短报告或小文章的 T_EX 格式.

statistik 提供各章的页数统计.

stdclsdv 为宏包编写者提供信息.

stdpage 给定理打阴影背景.

stmaryrd 圣玛丽道路 (St Mary Road) 符号字库.

struktex 生成 Nassi Shneidermann 结构图表.

sttools 利用 L^AT_EX 的内核命令写的几个宏包, 尤其适用于分栏的页面版式.

stubs 在页的下部重复打印一条条可以撕下来的联系方式信息, 供张贴用.

subeqn 对方程式组内部的方程编号.

subeqnarray 对内部等式编号的等式组.

subfig 取代较旧的 subfigure.

subfigure 给插图内部的子图编号.

subfiles 允许对主文件里 \input 的子文件单独编译.

subfloat 对插图和表格的内部成分编号.

substr 处理字符串内部的子串的命令.

sudoku 绘制 sudoku 猜谜游戏的框格.

supertabular 处理伸展多页的表格, 不过 longtable 更好用些.

svn 用于打印版本控制系统号.

svninfo 用于提取版本控制系统内的版本号及文件信息.

swebib 对标准 BibT_EX 格式的瑞典文版本.

swimgraf 用图形或文本显示游泳运动员的表现.

switcheml 颠倒电邮的地址使得他人不能窃取, 但打印出来是正确的.

swrule 画中间粗两端细的直线.

syntax 画句法图.

synttree 打印 Chomsky 的生成语法中使用的句法树.

t-angles 可以画辫子 Hopf 代数的运算以及其他图形.

t-rsteps ConT_EXt 中生成幻灯片用的包.

t1-fraktur T1 编码的 Fraktur 字体 (德文花体).

t2 T2 编码用到的一些文件.

tabbing 允许在制表位环境里使用带重音符号的字母.

- tabulary** 定义一个新的表格环境, 使得总宽度和某些列宽可被指定, 又能自动平衡各列的宽度.
- talk** 用于生成演示用的幻灯片或打印透明片, 它允许同时使用多种格式并能互相切换, 目前尚在开发更多的格式.
- tamefloats** 试图改善 L^AT_EX 安放浮动图表或脚注时可能产生的问题.
- tap** 用于打印复杂表格, 允许画斜线、阴影以及着色.
- tapir** 用于捷克文、斯洛伐克文和波兰文的一种字体.
- taupin** 可以重定义章节标题的字体以及小型大写字母.
- taylor** Taylor 开发的绘数学交换图的宏包.
- tbe** Arvind Borde 的书 “TeX by Example” 里的例子.
- tcldoc** 定义了一些环境及命令, 用于为 Tcl (工具命令语言) 的 .dtx 格式的源程序中插入说明.
- tds** T_EX 的目录结构文件.
- technics** 提供技术报告的模板.
- template** L^AT_EX3 里使用的模板.
- tengwar** 打印土耳其文的字库.
- tensor** 用于打印含有上下标以及前置指标的对象.
- testflow** 用于测试 PS 或 PDF 的打印情况.
- teubner** 用于打印希腊文.
- tex-ps** T_EX 到 PostScript 的宏包及附件, 包括 EPS 文件的变换、打印准备、分色、镜像变换等.
- tex-refs** T_EX 的参考资料.
- tex4ht** 用于把 L^AT_EX 文件转换成网页文件, 见本书第 14 章.
- texdraw** 图形包, 参见本书 §7.7.
- texlogos** 生成各种 L^AT_EX 的图标.
- texmate** 打印国际象棋谱.
- texpower** 为演示用 PDF 文件提供动态效果, 如换页、彩色突出显示等.
- texshade** 使核苷和缩氨酸对齐的包.
- texsis** 这是 PlainT_EX 的宏包, 提供 L^AT_EX 的许多功能.
- textcase** 改变其作用范围内字母的大小写, 但是不影响其中的数学公式以及 T_EX 命令.
- textfit** 通过改变字体的大小使得文本适应指定的宽度或高度.
- textopo** 设置带注解的膜蛋白拓扑图.
- textpos** 把盒子放在一页的绝对位置上.
- thesis** 学位论文的格式.
- thmbox** 为定理的叙述配置各种边框.
- threed** 用 MetaPost 创建 3 维物体 (例如多面体) 的动画.
- thumb** 把缩略记号放在文件里.
- thumbpdf** 利用 Perl 以及 GhostScript 生成缩略图, 供 pdfT_EX 以及 dvips/ps2pdf 使用.
- ticket** 用 L^AT_EX 生成标签、名片、备忘条等.
- timesht** 生成活动日程安排表.
- timetable** 生成时间表.
- timing** 电位等随时间变化的示意图.
- tipa** 国际音标的宏包与字库.

titlefoot 把各种内容添加到标题页的脚注。
titlesec 提供一个界面使你能设计自己的章节标题格式。
titling 提供对 `\maketitle` 命令的排版控制手段。
tocbibind 自动把参考文献表、索引等加入目录。
tocloft 提供对各类目录的排版控制。
tocvsec2 提供对章节编号的控制。
todo 在文件后面加上计划要作的事项 (to-do) 列表。
tokenizer 把一个字符串按逗号拆分成几个 token。
toolbox 供 T_EX 编程者用的工具包, 有多种功能。
tools 由 L^AT_EX3 项目组成员编写的 L^AT_EX 2_ε 的各种工具。
topfloat 把浮动元素放到页顶。
totpages 提供文件的总页数以及最后一页的页码。
tpslifonts 选择合适的字体使得投影仪演示得到更清晰的效果。
tracking 自动调节词或句内的符号间的空间, 以适合特定的长度。
trajan 罗马 Trajan 柱子 (公元 114 年) 上用的字体。
tree-dvips 画句法分析的树形图。
treesvr 画二叉或三叉树。
treetex 利用 C 程序画任意多个分叉以及节点的树形图。
trfsigns 打印 Laplace 变换、Fourier 变换等各种变换的符号。
trsym 打印 Laplace 变换、Fourier 变换等各种变换的符号。
ttf-MiKTeX 南开大学孙文昌开发的系列宏包: `ttfshape`, `pdftrick4win`, `picture`, `truetype`, `unilabel`, `texfriend` 等。
tugboat TUGboat 论文的 L^AT_EX 宏包。
twoup 在一张纸上打印两页。
txfonts Adobe Times (或 URW NimbusRomNo9L) 的 T_EX 符号的 Type 1 字库, 可用于 OT1, T1, TS1, LY1 编码。
type1cm 解除对 CM 字库的离散尺寸限制。
typedref 把引用细分成章节引用、插图引用等。
typespec 打印字库文件的样本以及有用信息。
typogrid 打印排印用的垂直线。
uaclasses 亚利桑那大学学位论文格式。
ucthesis 加利福尼亚大学学位论文格式。
ugq URW Grotesk Bold 的 PostScript 字体。
uhrzeit 日期与时间的格式。
uiucthesis 伊利诺斯大学学位论文格式。
ukrhyph 乌克兰文换行规则。
uk-tex-faq 英国的 T_EX 常用问题解答。
ulsy 额外的数学字符。
umlaute 使得超过 128 的 ASCII 码也能接受。
umoline 画允许换行的下划线。
umrand 提供各种形状的边框。
underbracket 画下划的括号。
underlin 用于画下划线。
undertilde 在数学符号下面划代字号 (tilde)。
uni 一种字体的 MetaFont 文件。

unicode 把 Unicode 输入转换成 L^AT_EX.
units 用于打印分式及物理单位.
unitsdef 用于排印单位.
universa Universal 字库及其支持文件.
unsupported Knuth 的 MetaFont 源文件.
upquote 打印直立的引号.
urlbst BibTeX 对网页的支持.
urwstd URW 的 PS 字库, 可以代替商业化的 Adobe 字体.
urwvn 加入了越南文字母的 URW 字库.
usenix USENIX (先进运算系统协会, The Advanced Computing Systems Association) 的科技论文宏包.
ushort 较短或较长的下划线.
ut-thesis Toronto 大学学位论文格式.
utopia Adobe 提供的 Utopia 字库.
utthesis Austin Texas 大学学位论文格式.
uwmslide Wisconsin-Madison 大学的投影片格式.
uwthesis Washington 大学学位论文格式.
vancouver 生物医学杂志用的参考文献格式.
varindex 索引命令的变形.
vdm 打印 VDM 格式的文件.
vector 方便向量的表示.
venn 画集合的文氏图的 MetaPost 包.
verse 用于打印诗歌.
versions 可以选择打印或不打印版本号.
vhistory 用于创建文件的版本演变历史纪录.
viking 为 Viking 文使用的 T_EX.
vita 为准备履历表而设计.
vmargin 方便改变页边空白的大小.
vnTEX 对越南文的支持.
volumes 用于把一个 T_EX 文件中的某一部分单挑出来打印, 但是仍然生成与此内容相关的目录、索引等附加内容.
vrB 抄录环境.
vrSion 为 DVI 文件加上版本号.
vtEX 对输出版面有更灵活的控制, 尤其是分成两栏的文本.
wallpaper 为 L^AT_EX 的输出加墙纸 (背景图形).
warning 把整体警告放在 log 文件的末尾.
warpcal 定义 tabular 环境的一个新的列格式, 使得数字可以右边对齐.
was 含几个文件, 如打印直立希腊字母, 数学黑体等.
wasy Waldis 符号字库.
wasysym Waldis 符号字库的附加字符, 对于使用宏包 **amsfonts** 或 **amssymb** 的用户, 这个包的用处不太大.
webeq 通过 L^AT_EX 的输入得到 PDF 输出, 以获得教育用电子出版物.
webomints 提供一些边框装饰图案.
whuthesis 黄正华开发的武汉大学博硕士学位论文宏包.
williams P. Williams 的各种宏包.

withesis Wisconsin-Madison 大学的学位论文格式.
wnri 古英语、印度语和印第安语的一些 MetaFont 字库.
wntamil 对泰米尔语的支持.
wordcount 估计 L^AT_EX 源文件中的词汇数.
wp-conv 能把 L^AT_EX 源文件与 RTF、Word、WordPerfect 文件之间互相转换的宏包列表.
wrapfig 使文本能在图形周围绕排.
wsuipa 使用国际音标的格式与字库.
xcolor 与驱动器无关的彩色工具.
xdoc 对 T_EX 文件系统的扩展.
xfrac 打印 1/2 样式的分式.
xinitials L^AT_EX3 的包.
xjtuthesis 李树钧开发的西安交通大学博士学位论文宏包.
xkeyval keyval 包的扩展.
xlop 可以用竖式或其他适合教学用的格式打印算术公式.
xmlplay XML_{TeX} 的实例, 莎士比亚戏剧的排版.
xmltex 用 T_EX 来排版 XML 文件.
xmpincl 把 XML 文件中的元数据嵌入文件.
xor L^AT_EX3 的包.
xparse L^AT_EX3 的包.
xtab supertabular 的扩展, 支持表格分页.
xtcpts 用于定义使用多种语言的标题.
xtheorem L^AT_EX3 的包.
xymtex 排印化学结构.
xypic 绘图宏包.
yafoot 各种脚注命令.
yannisgr 一些希腊字库.
yfonts 由 Yannis Haralambous 设计的古德文字库, 包括 Gotisch, Fraktur, Schwabacher 等.
yhmath L^AT_EX 的扩展数学字库.
yi4latex 对彝族文字的支持.
york-thesis York 大学的学位论文格式.
youngtab 用于打印表示论里的杨图.
yplan 带记事表的日历.
ytex MIT 开发的 T_EX 的宏 (不同于 L^AT_EX).
zed-csp 排印 Z 与 CSP 格式.
zefonts 用于模拟 dc 字库.

附录五 L^AT_EX 能使用的符号

随书光盘的 Doc\常用英文文档\The Comprehensive L^AT_EX Symbol List (20050922).pdf 收录了 L^AT_EX 可以自由使用的各种符号 3300 种, 我们这里精选了一部分, 供读者参考. 注意许多符号只能以 PS 的形式打印或显示, 有的要先导入相应的宏包, 例如 \mathcal{A} 符号应该先导入 amssymb 才能使用.

第一部分 正文部分使用的符号

表 1 L^AT_EX 中已定义的文本格式符号

^	\textasciicircum		<	\textless
~	\textasciitilde	a	ˆ	\textordfeminine
*	\textasteriskcentered	o	Ω	\textordmasculine
\	\textbackslash		¶	\textparagraph
	\textbar		.	\textperiodcentered
{	\textbraceleft		¿	\textquestiondown
}	\textbraceright		“	\textquotedblleft
•	\textbullet		”	\textquotedblright
©	\textcopyright		‘	\textquoteleft
†	\textdagger		’	\textquoteright
‡	\textdaggerdbl	®	®	\textregistered
\$	\textdollar		§	\textsection
...	\textellipsis		£	\textsterling
—	\textemdash	™	™	\texttrademark
-	\textendash		-	\textunderscore
!	\textexclamdown		~	\textvisiblespace
>	\textgreater			

当出现两个符号时, 左边的是 L^AT_EX 的默认输出, 右边的是使用宏包 textcomp 后的输出.

表 2 textcomp 提供的货币单位

₮	\textbaht	\$	\textdollaroldstyle	₪	\textnaira
¢	\textcent	₫	\textdong	₭	\textpeso
¢	\textcentoldstyle	€	\texteuro	£	\textsterling
₯	\textcolonmonetary	f	\textflorin	₩	\textwon
₡	\textcurrency	₲	\textguarani	¥	\textyen
\$	\textdollar	₱	\textlira		

表3 marvosym 提供的货币单位

₴	\Denarius	€	\EURcr	€	\EURtm	β	\Shilling
@	\Ecommerce	€	\EURdig	\$	\EyesDollar		
€	\EUR	€	\EURhv	ℳ	\Pfund		

为了与不同的字模配合, 欧元记号有几种: Courier (\EURcr), Helvetica (\EURhv), Times (\EURtm), 以及配合 marvosym 数字 (表 93) 的 \EURdig.

表4 wasysym 提供的货币单位

¢	\cent	¤	\currency
---	-------	---	-----------

表5 eurosym 提供的欧元符号

€	\geneuro	€	\geneurowide	€	\officiaeuro
€	\geneuronarrow				

表6 textcomp 的法定符号

⒫	\textcircledP	©	©	\textcopyright
Ⓒ	\textcircleft	®	®	\textregistered
SM	\textservicemark	TM	TM	\texttrademark

当出现两个符号时, 左边的是 L^AT_EX 的默认输出, 右边的是使用宏包 textcomp 后的输出.

表7 ccllicenses 的 Creative Commons 版权图标

CC	\cc	NC	\ccnc	SA	\ccsa
BY	\ccby	ND	\ccnd		

表8 textcomp 的其他符号

*	\textasteriskcentered	o	\textopenbullet
	\textbardbl	a a	\textordfeminine
○	\textbigcircle	o Ω	\textordmasculine
␣	\textblank	¶	\textparagraph
	\textbrokenbar	.	\textperiodcentered
•	\textbullet	‰	\textpertenthousand
†	\textdagger	‰	\textperthousand
‡	\textdaggerdbl	¶	\textpilcrow
=	\textdblhyphen	'	\textquotesingle
=	\textdblhyphenchar	,	\textquotestraightbase
%	\textdiscount	"	\textquotestraightdblbase
€	\textestimated	R	\textrecipe
?	\textinterrobang	※	\textreferencemark
‡	\textinterrobangdown	§	\textsection
♪	\textmusicalnote	—	\textthreequartersemdash
Nº	\textnumero	—	\texttwelveudash

当出现两个符号时, 左边的是 L^AT_EX 的默认输出, 右边的是使用宏包 textcomp 后的输出.

表9 textcomp的古体数字

0 \textzerooldstyle	4 \textfouroldstyle	8 \texteightoldstyle
1 \textoneoldstyle	5 \textfiveoldstyle	9 \textnineoldstyle
2 \texttwooldstyle	6 \textsixoldstyle	
3 \textthreeoldstyle	7 \textsevenoldstyle	

也可使用 \oldstylenums{..} 来打印古体数字.

第二部分 数学符号

表10 二元运算符

∩ \amalg	∪ \cup	⊕ \oplus	× \times
* \ast	† \dagger	⊗ \otimes	◀ \triangleleft
○ \bigcirc	‡ \ddagger	⊗ \otimes	▶ \triangleright
▽ \bigtriangledown	◇ \diamond	± \pm	≤ \leq
△ \bigtriangleup	÷ \div	▷ \rhd	≥ \geq
• \bullet	◁ \lhd	\ \setminus	⊕ \oplus
∩ \cap	≠ \neq	□ \sqcap	∨ \vee
· \cdot	○ \odot	□ \sqcup	∧ \wedge
○ \circ	⊖ \ominus	★ \star	ℳ \mathcal{M}

* 在 L^AT_EX 内没有定义, 必须调入以下宏包之一: latexsym, amsfonts, amssymb, txfonts, pxfonts, wasysym.

表11 AMS的二元运算符

⌋ \barwedge	⊙ \circledcirc	⌈ \intercal
⊠ \boxdot	⊖ \circleddash	↗ \leftthreetimes
⊞ \boxminus	⊔ \Cup	⋈ \ltimes
⊕ \boxplus	∨ \curlyvee	↘ \rightthreetimes
⊗ \boxtimes	∧ \curlywedge	⋉ \rtimes
⌒ \Cap	⋈ \divideontimes	↖ \smallsetminus
· \centerdot	⊕ \dotplus	∨ \veebar
⊗ \circledast	⌋ \doublebarwedge	

表12 txfonts/pxfonts的二元运算符

⊙ \circledbar	⊙ \circledwedge	○ \medcirc
⊙ \circledbslash	⌘ \invamp	⊞ \sqcapplus
⊙ \circledvee	● \medbullet	⊞ \sqcupplus

表 13 stmaryrd 的二元运算符

ϕ \baro	\parallel \interleave	\otimes \varoast
\parallel \bbslash	\triangleleft \leftslice	\oplus \varobar
$\&$ \binampersand	\M \merge	\oslash \varobslash
\wp \bindnasrepma	\ominus \minuso	\odot \varocircle
\boxminus \boxast	\pm \moo	\odot \varodot
\boxplus \boxbar	\oplus \nplus	\oslash \varogreaterthan
\boxtimes \boxbox	\oplus \obar	\oslash \varolessthan
\boxdiv \boxbslash	\square \oblong	\ominus \varominus
\boxcirc \boxcircle	\oslash \obslash	\oplus \varoplus
\boxdot \boxdot	\oslash \ogreaterthan	\oslash \varoslash
\square \boxempty	\oslash \olessthan	\otimes \varotimes
\boxtimes \boxslash	\oslash \ovee	\oslash \varovee
\curlyvee \curlyveedownarrow	\oslash \owedge	\oslash \varowedge
\curlyvee \curlyveeuparrow	\triangleright \rightslice	\times \vartimes
\curlywedge \curlywedgedownarrow	\parallel \sslash	γ \Ydown
\curlywedge \curlywedgeuparrow	\parallel \talloblong	\prec \Yleft
\backslash \fatbslash	\bigcirc \varbigcirc	\succ \Yright
$;$ \fatsemi	γ \varcurlyvee	γ \Yup
\parallel \fatslash	γ \varcurlywedge	

表 14 mathabx 的二元运算符

$*$ \ast	\wedge \curlywedge	\sqcap \sqcap
$*$ \Asterisk	\div \divdot	\sqcup \sqcup
\wedge \barwedge	$*$ \divideontimes	\sqcap \sqdoublecap
\star \bigstar	\div \dotdiv	\sqcup \sqdoublecup
\star \bigvarstar	\dagger \dotplus	\square \square
\blacklozenge \blackdiamond	\times \dottimes	\boxplus \sqplus
\cap \cap	\barwedge \doublebarwedge	\cdot \udot
\dagger \circplus	\cap \doublecap	\oplus \uplus
$*$ \coasterisk	\cup \doublecup	$*$ \varstar
$*$ \coAsterisk	\ltimes \ltimes	\vee \vee
$*$ \convolution	\oplus \pluscirc	\veebar \veebar
\cup \cup	\rtimes \rtimes	\leq \veedoublebar
\vee \curlyvee	\blacksquare \sqbullet	\wedge \wedge

表 15 wasysym 的二元运算符

\triangleleft \lhd	\bigcirc \ocircle	\blacktriangleright \RHD	\supseteq \unrhd
\blacktriangleleft \LHD	\triangleright \rhd	\sqsubseteq \unlhd	

表 16 ulsy 的几何二元运算符

\oplus \odplus

表 17 mathabx 的几何二元运算符

\blacktriangledown \blacktriangledown	\boxright \boxright	\ominus \ominus
\blacktriangleleft \blacktriangleleft	\boxslash \boxslash	\oplus \oplus
\blacktriangleright \blacktriangleright	\boxtimes \boxtimes	\otimes \otimes
\blacktriangleup \blacktriangleup	\boxtop \boxtop	\oslash \oslash
\boxast \boxast	\boxtriangleup \boxtriangleup	\otimes \otimes
\boxbackslash \boxbackslash	\boxvoid \boxvoid	\otop \otop
\boxbot \boxbot	\oasterisk \oasterisk	\otriangleup \otriangleup
\boxcirc \boxcirc	\oboxbackslash \oboxbackslash	\ovoid \ovoid
\boxcoasterisk \boxcoasterisk	\obot \obot	\smalltriangledown \smalltriangledown
\boxdiv \boxdiv	\ocirc \ocirc	\smalltriangleleft \smalltriangleleft
\boxdot \boxdot	\occoasterisk \occoasterisk	\smalltriangleright \smalltriangleright
\boxleft \boxleft	\odiv \odiv	\smalltriangleup \smalltriangleup
\boxminus \boxminus	\odot \odot	
\boxplus \boxplus	\oleft \oleft	

表 18 大小可变的运算符

\cap \bigcap	\sqcup \bigsqcup	\int \int
\cup \bigcup	\uplus \biguplus	\oint \oint
\odot \bigodot	\vee \bigvee	\prod \prod
\oplus \bigoplus	\wedge \bigwedge	\sum \sum
\otimes \bigotimes	\coprod \coprod	

表 19 AMS 的大小可变的运算符

\iint \iint	\iiint \iiint
\iiint \iiint	$\int \dots \int$ \idotsint

表 20 wasysym 的大小可变的运算符

\iiint \iiint	\oint \oint	\oint \varoint
\iint \iint	\int \int	

表 21 stmaryrd 的大小可变的运算符

$\square\square$ \bigbox	$\parallel\parallel$ \biginterleave	$\sqcap\sqcap$ \bigsqcap
\curlyvee \bigcurlyvee	$\oplus\oplus$ \bignplus	$\nabla\nabla$ \bigtriangledown
\curlywedge \bigcurlywedge	$\parallel\parallel$ \bigparallel	$\triangle\triangle$ \bigtriangleup

表 22 mathabx 的大小可变的运算符

\curlyvee \bigcurlyvee	\boxslash \bigboxslash	$\oplus\oplus$ \bigorright
$\sqcap\sqcap$ \bigsqcap	\boxtimes \bigboxtimes	$\oslash\oslash$ \bigoslash
\curlywedge \bigcurlywedge	\boxplus \bigboxplus	$\oplus\oplus$ \bigotop
\boxast \bigboxasterisk	$\triangle\triangle$ \bigboxtriangleup	$\triangle\triangle$ \bigotriangleup
\boxbackslash \bigboxbackslash	$\square\square$ \bigboxvoid	$\bigcirc\bigcirc$ \bigovoid
\boxbot \bigboxbot	$\complement\complement$ \bigcomplementtop	$++$ \bigplus
\boxcirc \bigboxcirc	\boxast \bigboxasterisk	\boxplus \bigboxplus
\boxcoasterisk \bigboxcoasterisk	$\oslash\oslash$ \bigobackslash	$\times\times$ \bigtimes
\boxdiv \bigboxdiv	$\oplus\oplus$ \bigobot	\iiint \iiint
\boxdot \bigboxdot	$\odot\odot$ \bigocirc	\iint \iint
\boxleft \bigboxleft	\boxcoasterisk \bigboxcoasterisk	\int \int
\boxminus \bigboxminus	$\div\div$ \bigodiv	\oiint \oiint
\boxplus \bigboxplus	$\oplus\oplus$ \bigoleft	\oint \oint
\boxright \bigboxright	$\ominus\ominus$ \bigominus	

表 23 txfonts/pxfonts 的大小可变的运算符

\boxplus	\boxplus	<code>\bigsqcapplus</code>	\oint	\oint	<code>\ointclockwise</code>
\boxcup	\boxcup	<code>\bigsqcupplus</code>	\oint	\oint	<code>\ointctrackwise</code>
f	f	<code>\fint</code>	\int	\int	<code>\sqiiint</code>
$\int \dots \int$	$\int \dots \int$	<code>\idotsint</code>	\int	\int	<code>\sqiint</code>
\iiint	\iiint	<code>\iiiint</code>	\int	\int	<code>\sqint</code>
\iiint	\iiint	<code>\iiint</code>	\int	\int	<code>\varoiintclockwise</code>
\iint	\iint	<code>\iint</code>	\int	\int	<code>\varoiintctrackwise</code>
\oiint	\oiint	<code>\oiintclockwise</code>	\oint	\oint	<code>\varoiintclockwise</code>
\oiint	\oiint	<code>\oiintctrackwise</code>	\oint	\oint	<code>\varoiintctrackwise</code>
\oiint	\oiint	<code>\oiint</code>	\oint	\oint	<code>\varointclockwise</code>
\oint	\oint	<code>\ointclockwise</code>	\oint	\oint	<code>\varointctrackwise</code>
\oint	\oint	<code>\ointctrackwise</code>	\times	\times	<code>\varprod</code>
\oint	\oint	<code>\oint</code>			

表 24 esint 的大小可变的运算符

$\int \dots \int$	$\int \dots \int$	<code>\dotsint</code>	\oint	\oint	<code>\ointclockwise</code>
f	f	<code>\fint</code>	\oint	\oint	<code>\ointctrackwise</code>
\iiint	\iiint	<code>\iiiint</code>	\int	\int	<code>\sqint</code>
\iiint	\iiint	<code>\iiint</code>	\int	\int	<code>\sqint</code>
\iint	\iint	<code>\iint</code>	\int	\int	<code>\varoiint</code>
\int	\int	<code>\landdownint</code>	\oint	\oint	<code>\varointclockwise</code>
\int	\int	<code>\landupint</code>	\oint	\oint	<code>\varointctrackwise</code>
\oint	\oint	<code>\oint</code>			

表 25 二元关系符

\approx	<code>\approx</code>	\equiv	<code>\equiv</code>	\perp	<code>\perp</code>	\smile	<code>\smile</code>
\asymp	<code>\asymp</code>	\frown	<code>\frown</code>	\prec	<code>\prec</code>	\succ	<code>\succ</code>
\bowtie	<code>\bowtie</code>	\Join^*	<code>\Join^*</code>	\preceq	<code>\preceq</code>	\succeq	<code>\succeq</code>
\cong	<code>\cong</code>	\mid	<code>\mid</code>	\propto	<code>\propto</code>	\vdash	<code>\vdash</code>
\dashv	<code>\dashv</code>	\models	<code>\models</code>	\sim	<code>\sim</code>		
\doteq	<code>\doteq</code>	\parallel	<code>\parallel</code>	\simeq	<code>\simeq</code>		

* 在 L^AT_EX 内没有定义, 必须调入以下宏包之一: latexsym, amsfonts, amssymb, mathabx, txfonts, pxfonts, wasysym.

表 26 AMS 二元关系符

\approx	<code>\approxeq</code>	\equiv	<code>\eqcirc</code>	\succapprox	<code>\succapprox</code>
\backepsilon	<code>\backepsilon</code>	\fallingdotseq	<code>\fallingdotseq</code>	\succcurlyeq	<code>\succcurlyeq</code>
\backsim	<code>\backsim</code>	\multimap	<code>\multimap</code>	\succsim	<code>\succsim</code>
\backsimeq	<code>\backsimeq</code>	\pitchfork	<code>\pitchfork</code>	\therefore	<code>\therefore</code>
\because	<code>\because</code>	\precapprox	<code>\precapprox</code>	\thickapprox	<code>\thickapprox</code>
\between	<code>\between</code>	\preccurlyeq	<code>\preccurlyeq</code>	\thicksim	<code>\thicksim</code>
\bumpeq	<code>\bumpeq</code>	$\prec sim$	<code>\prec sim</code>	\varpropto	<code>\varpropto</code>
\bumpeq	<code>\bumpeq</code>	\risingdotseq	<code>\risingdotseq</code>	\Vdash	<code>\Vdash</code>
\circeq	<code>\circeq</code>	\shortmid	<code>\shortmid</code>	\vdash	<code>\vdash</code>
\curlyeqprec	<code>\curlyeqprec</code>	\shortparallel	<code>\shortparallel</code>	\Vdash	<code>\Vdash</code>
\curlyeqsucc	<code>\curlyeqsucc</code>	\smallfrown	<code>\smallfrown</code>		
\doteqdot	<code>\doteqdot</code>	\smallsmile	<code>\smallsmile</code>		

表 27 AMS 否定二元关系符

\ncong	<code>\ncong</code>	\nshortparallel	<code>\nshortparallel</code>	\nVDash	<code>\nVDash</code>
\nmid	<code>\nmid</code>	\nsim	<code>\nsim</code>	\precnapprox	<code>\precnapprox</code>
\nparallel	<code>\nparallel</code>	\nsucc	<code>\nsucc</code>	\precnsim	<code>\precnsim</code>
\nprec	<code>\nprec</code>	\nsucceq	<code>\nsucceq</code>	\succnapprox	<code>\succnapprox</code>
\npreceq	<code>\npreceq</code>	\nvDash	<code>\nvDash</code>	\succnsim	<code>\succnsim</code>
\nshortmid	<code>\nshortmid</code>	\nvdash	<code>\nvdash</code>		

表 28 stmaryrd 二元关系符

\inplus	<code>\inplus</code>	\niplus	<code>\niplus</code>
-----------	----------------------	-----------	----------------------

表 29 wasysym 二元关系符

\invneg	<code>\invneg</code>	\leadsto	<code>\leadsto</code>	\wasympropto	<code>\wasympropto</code>
\Join	<code>\Join</code>	\logof	<code>\logof</code>		

表 30 txfonts/pxfonts 二元关系符

\odot	<code>\circledgtr</code>	\ltimes	<code>\lJoin</code>	\times	<code>\opentimes</code>
\ominus	<code>\circledless</code>	\rtimes	<code>\lrtimes</code>	\perp	<code>\Perp</code>
\approx	<code>\colonapprox</code>	\multimap	<code>\multimap</code>	\preceq	<code>\preceqq</code>
\Colonapprox	<code>\Colonapprox</code>	\multimap	<code>\multimapboth</code>	\preceq	<code>\precneqq</code>
\coloneq	<code>\coloneq</code>	\multimap	<code>\multimapbothvert</code>	\Join	<code>\rJoin</code>
\Coloneq	<code>\Coloneq</code>	\multimap	<code>\multimapdot</code>	\strictfi	<code>\strictfi</code>
\coloneqq	<code>\coloneqq</code>	\multimap	<code>\multimapdotboth</code>	\strictif	<code>\strictif</code>
\Coloneqq	<code>\Coloneqq</code>	\multimap	<code>\multimapdotbothA</code>	\strictiff	<code>\strictiff</code>
\colonsim	<code>\colonsim</code>	\multimap	<code>\multimapdotbothAvert</code>	\succeq	<code>\succeqq</code>
\Colonsim	<code>\Colonsim</code>	\multimap	<code>\multimapdotbothB</code>	\succneqq	<code>\succneqq</code>
\Eqcolon	<code>\Eqcolon</code>	\multimap	<code>\multimapdotbothBvert</code>	\parallel	<code>\varparallel</code>
\eqcolon	<code>\eqcolon</code>	\multimap	<code>\multimapdotbothvert</code>	\parallel	<code>\varparallelinv</code>
\eqqcolon	<code>\eqqcolon</code>	\multimap	<code>\multimapdotinv</code>	\Vdash	<code>\Vdash</code>
\Eqqqcolon	<code>\Eqqqcolon</code>	\multimap	<code>\multimapinv</code>		
\eqsim	<code>\eqsim</code>	\times	<code>\openJoin</code>		

表 31 txfonts/pxfonts 否定二元关系符

\napprox	<code>\napproxeq</code>	\nprec	<code>\npreccurlyeq</code>	\nthickapprox	<code>\nthickapprox</code>
\nasympt	<code>\nasympt</code>	\npreceq	<code>\npreceqq</code>	\twoheadleftarrow	<code>\twoheadleftarrow</code>
\nbacksim	<code>\nbacksim</code>	\nprec	<code>\nprec</code>	\twoheadrightarrow	<code>\twoheadrightarrow</code>
\nbacksimeq	<code>\nbacksimeq</code>	\nsimeq	<code>\nsimeq</code>	\nvarparallel	<code>\nvarparallel</code>
\nbumpeq	<code>\nbumpeq</code>	\nsuccapprox	<code>\nsuccapprox</code>	\nvarparallelinv	<code>\nvarparallelinv</code>
\nBumpeq	<code>\nBumpeq</code>	\nsucccurlyeq	<code>\nsucccurlyeq</code>	\nVdash	<code>\nVdash</code>
\nequiv	<code>\nequiv</code>	\nsucceq	<code>\nsucceqq</code>		
\nprecapprox	<code>\nprecapprox</code>	\nsuccsim	<code>\nsuccsim</code>		

表 32 mathabx 二元关系符

\between	<code>\between</code>	\divides	<code>\divides</code>	\risingdotseq	<code>\risingdotseq</code>
\botdoteq	<code>\botdoteq</code>	\dotseq	<code>\dotseq</code>	\succapprox	<code>\succapprox</code>
\Bumpedeq	<code>\Bumpedeq</code>	\eqbumped	<code>\eqbumped</code>	\succcurlyeq	<code>\succcurlyeq</code>
\bumpedeq	<code>\bumpedeq</code>	\eqcirc	<code>\eqcirc</code>	\succdot	<code>\succdot</code>
\circeq	<code>\circeq</code>	\eqcolon	<code>\eqcolon</code>	\succsim	<code>\succsim</code>
\coloneq	<code>\coloneq</code>	\fallingdotseq	<code>\fallingdotseq</code>	\therefore	<code>\therefore</code>
\corresponds	<code>\corresponds</code>	\ggcurly	<code>\ggcurly</code>	\topdoteq	<code>\topdoteq</code>
\curlyeqprec	<code>\curlyeqprec</code>	\llcurly	<code>\llcurly</code>	\vdash	<code>\vdash</code>
\curlyeqsucc	<code>\curlyeqsucc</code>	\precapprox	<code>\precapprox</code>	\Vdash	<code>\Vdash</code>
\DashV	<code>\DashV</code>	\preccurlyeq	<code>\preccurlyeq</code>	\Vdash	<code>\Vdash</code>
\Dashv	<code>\Dashv</code>	\precdot	<code>\precdot</code>	\Vdash	<code>\Vdash</code>
\dashVv	<code>\dashVv</code>	\prec	<code>\prec</code>		

表 33 mathabx 否定二元关系符

\approx	<code>\napprox</code>	\perp	<code>\notperp</code>	\nVdash	<code>\nvDash</code>
\cong	<code>\ncong</code>	\nprec	<code>\nprec</code>	\nVDash	<code>\nVDash</code>
\curlyeqprec	<code>\ncurlyeqprec</code>	\nprecapprox	<code>\nprecapprox</code>	\nVdash	<code>\nVdash</code>
\curlyeqsucc	<code>\ncurlyeqsucc</code>	\npreccurlyeq	<code>\npreccurlyeq</code>	\nvdash	<code>\nvdash</code>
\nDashv	<code>\nDashv</code>	\npreceq	<code>\npreceq</code>	\nVdash	<code>\nVdash</code>
\ndashV	<code>\ndashV</code>	\nprecsim	<code>\nprecsim</code>	\precnapprox	<code>\precnapprox</code>
\ndashv	<code>\ndashv</code>	\nsim	<code>\nsim</code>	\precneq	<code>\precneq</code>
\nDashV	<code>\nDashV</code>	\nsimeq	<code>\nsimeq</code>	\precnsim	<code>\precnsim</code>
\ndashVv	<code>\ndashVv</code>	\nsucc	<code>\nsucc</code>	\succnapprox	<code>\succnapprox</code>
\neq	<code>\neq</code>	\nsuccapprox	<code>\nsuccapprox</code>	\succneq	<code>\succneq</code>
\notasym	<code>\notasym</code>	\nsucccurlyeq	<code>\nsucccurlyeq</code>	\succnsim	<code>\succnsim</code>
\notdivides	<code>\notdivides</code>	\nsucceq	<code>\nsucceq</code>		
\notequiv	<code>\notequiv</code>	\nsucssim	<code>\nsucssim</code>		

命令 `\changenotsign` 可以切换否定命令 `\not` 的形状: 使它成为竖线或右下斜线。

表 34 trsym 二元关系符

$\circ \dashrightarrow$	<code>\InversTransformHoriz</code>	$\dashrightarrow \circ$	<code>\TransformHoriz</code>
$\circ \dashv$	<code>\InversTransformVert</code>	$\dashv \circ$	<code>\TransformVert</code>

表 35 trfsigns 二元关系符

\curvearrowright	<code>\dfourier</code>	\curvearrowleft	<code>\Dfourier</code>
\frown	<code>\fourier</code>	\frown	<code>\Fourier</code>
$\circ \dashrightarrow$	<code>\laplace</code>	$\dashrightarrow \circ$	<code>\Laplace</code>
$\circ \dashv$	<code>\ztransf</code>	$\dashv \circ$	<code>\Ztransf</code>

表 36 集合包含关系符

\sqsubset	<code>\sqsubset*</code>	\sqsupseteq	<code>\sqsupseteq</code>	\supset	<code>\supset</code>
\sqsubseteq	<code>\sqsubseteq</code>	\subset	<code>\subset</code>	\supseteq	<code>\supseteq</code>
\sqsupset	<code>\sqsupset*</code>	\subseteq	<code>\subseteq</code>		

* 在 L^AT_EX 内没有定义, 必须调入以下宏包之一: `latexsym`, `amsfonts`, `amssymb`, `mathabx`, `txfonts`, `pxfonts`, `wasysym`.

表 37 AMS 集合包含关系符

\nsubseteq	<code>\nsubseteq</code>	\subseteqeq	<code>\subseteqeq</code>	\supsetneqq	<code>\supsetneqq</code>
\nsupseteq	<code>\nsupseteq</code>	\subsetneq	<code>\subsetneq</code>	\varsubsetneq	<code>\varsubsetneq</code>
\nsupseteqq	<code>\nsupseteqq</code>	\subsetneqq	<code>\subsetneqq</code>	\varsubsetneqq	<code>\varsubsetneqq</code>
\sqsubset	<code>\sqsubset</code>	\Supset	<code>\Supset</code>	\varsupsetneq	<code>\varsupsetneq</code>
\sqsupset	<code>\sqsupset</code>	\supseteqeq	<code>\supseteqeq</code>	\varsupsetneqq	<code>\varsupsetneqq</code>
\Subset	<code>\Subset</code>	\supsetneq	<code>\supsetneq</code>		

表 38 stmaryrd 集合包含关系符

\Subset	<code>\subsetplus</code>	\supset	<code>\supsetplus</code>
\subseteq	<code>\subsetpluseq</code>	\supseteq	<code>\supsetpluseq</code>

表 39 wasysym 集合包含关系符

\sqsubset	<code>\sqsubset</code>	\sqsupset	<code>\sqsupset</code>
-------------	------------------------	-------------	------------------------

表 40 txfonts/prfonts 集合包含关系符

\nsqsubset	<code>\nsqsubset</code>	\nsqsupseteq	<code>\nsqsupseteq</code>	\nsubset	<code>\nsubset</code>
\nsqsubseteq	<code>\nsqsubseteq</code>	\nsubseteq	<code>\nsubseteq</code>	\nsubset	<code>\nsubset</code>
\nsqsupset	<code>\nsqsupset</code>	\nsubseteqeq	<code>\nsubseteqeq</code>		

表 41 mathabx 集合包含关系符

\nsqsubset	<code>\nsqsubset</code>	\sqsubset	<code>\sqsubset</code>	\subsetneq	<code>\subsetneq</code>
\nsqSubset	<code>\nsqSubset</code>	\sqSubset	<code>\sqSubset</code>	\subsetneqq	<code>\subsetneqq</code>
\nsqsubseteq	<code>\nsqsubseteq</code>	\sqsubseteq	<code>\sqsubseteq</code>	\supset	<code>\supset</code>
\nsqsubseteqeq	<code>\nsqsubseteqeq</code>	\sqsubseteqeq	<code>\sqsubseteqeq</code>	\Supset	<code>\Supset</code>
\nsqsupset	<code>\nsqsupset</code>	\sqsupsetneq	<code>\sqsupsetneq</code>	\supseteq	<code>\supseteq</code>
\nsqSupset	<code>\nsqSupset</code>	\sqsupsetneqq	<code>\sqsupsetneqq</code>	\supseteqeq	<code>\supseteqeq</code>
\nsqsupseteq	<code>\nsqsupseteq</code>	\sqSupset	<code>\sqSupset</code>	\supsetneq	<code>\supsetneq</code>
\nsqsupseteqeq	<code>\nsqsupseteqeq</code>	\sqsupseteq	<code>\sqsupseteq</code>	\supsetneqq	<code>\supsetneqq</code>
\nsubset	<code>\nsubset</code>	\sqsupseteqeq	<code>\sqsupseteqeq</code>	\varsubsetneq	<code>\varsubsetneq</code>
\nSubset	<code>\nSubset</code>	\sqsupseteqeqq	<code>\sqsupseteqeqq</code>	\varsubsetneqq	<code>\varsubsetneqq</code>
\nsubseteq	<code>\nsubseteq</code>	\sqsupseteqneq	<code>\sqsupseteqneq</code>	\varsubsetneqq	<code>\varsubsetneqq</code>
\nsubseteqeq	<code>\nsubseteqeq</code>	\sqsupseteqneqq	<code>\sqsupseteqneqq</code>	\varsubsetneqq	<code>\varsubsetneqq</code>
\nsubsetneq	<code>\nsubsetneq</code>	\subset	<code>\subset</code>	\varsubsetneq	<code>\varsubsetneq</code>
\nSupset	<code>\nSupset</code>	\Subset	<code>\Subset</code>	\varsubsetneqq	<code>\varsubsetneqq</code>
\nsupseteq	<code>\nsupseteq</code>	\subseteq	<code>\subseteq</code>	\varsubsetneqq	<code>\varsubsetneqq</code>
\nsupseteqeq	<code>\nsupseteqeq</code>	\subseteqeq	<code>\subseteqeq</code>	\varsubsetneqq	<code>\varsubsetneqq</code>

表 42 不等号

\geq	<code>\geq</code>	\gg	<code>\gg</code>	\leq	<code>\leq</code>	\ll	<code>\ll</code>	\neq	<code>\neq</code>
--------	-------------------	-------	------------------	--------	-------------------	-------	------------------	--------	-------------------

表 43 wasysym 不等号

\gtrsim	<code>\apprge</code>	\lesssim	<code>\apprle</code>
-----------	----------------------	------------	----------------------

表 44 AMS 不等号

\gtrsim	<code>\eqslantgtr</code>	\gtrdot	<code>\gtrdot</code>	\lesseqgtr	<code>\ngeq</code>
\lessgtr	<code>\eqslantless</code>	\gtreqless	<code>\gtreqless</code>	\lesseqqgtr	<code>\ngeqq</code>
\geq	<code>\geqq</code>	\gtreqqlless	<code>\gtreqqlless</code>	\lessgtr	<code>\ngeqslant</code>
\gtrsim	<code>\geqslant</code>	\gtrless	<code>\gtrless</code>	\lessssim	<code>\ngtr</code>
\gg	<code>\ggg</code>	\gtrsim	<code>\gtrsim</code>	\lll	<code>\nleq</code>
\gtrapprox	<code>\gnapprox</code>	\gvertneqq	<code>\gvertneqq</code>	\lnapprox	<code>\nleqq</code>
\gneq	<code>\gneq</code>	\leqq	<code>\leqq</code>	\lneq	<code>\nleqslant</code>
\gneqq	<code>\gneqq</code>	\leqslant	<code>\leqslant</code>	\lneqq	<code>\nless</code>
\gnsim	<code>\gnsim</code>	\lessapprox	<code>\lessapprox</code>	\lnsim	
\gtrapprox	<code>\gtrapprox</code>	\lessdot	<code>\lessdot</code>	\lvertneqq	

表 45 txfonts/pxfonts 不等号

\ngg	<code>\ngg</code>	\ngtrsim	<code>\ngtrsim</code>	\nlessssim	<code>\nlessssim</code>
\ngtrapprox	<code>\ngtrapprox</code>	\nlessapprox	<code>\nlessapprox</code>	\nll	<code>\nll</code>
\ngtrless	<code>\ngtrless</code>	\nlessgtr	<code>\nlessgtr</code>		

表 46 mathabx 不等号

\gtrsim	<code>\eqslantgtr</code>	\gtreqless	<code>\gtreqless</code>	\lessssim	<code>\ngtr</code>
\lessgtr	<code>\eqslantless</code>	\gtreqqlless	<code>\gtreqqlless</code>	\ll	<code>\ngtrapprox</code>
\geq	<code>\geq</code>	\gtrless	<code>\gtrless</code>	\lll	<code>\ngtrsim</code>
\geqq	<code>\geqq</code>	\gtrsim	<code>\gtrsim</code>	\lnapprox	<code>\nleq</code>
\gg	<code>\gg</code>	\gvertneqq	<code>\gvertneqq</code>	\lneq	<code>\nleqq</code>
\ggg	<code>\ggg</code>	\leq	<code>\leq</code>	\lneqq	<code>\nless</code>
\gnapprox	<code>\gnapprox</code>	\leqq	<code>\leqq</code>	\lnsim	<code>\nlessapprox</code>
\gneq	<code>\gneq</code>	\lessapprox	<code>\lessapprox</code>	\lvertneqq	<code>\nlessssim</code>
\gneqq	<code>\gneqq</code>	\lessdot	<code>\lessdot</code>	\neqslantgtr	<code>\nvargeq</code>
\gnsim	<code>\gnsim</code>	\lesseqgtr	<code>\lesseqgtr</code>	\neqslantless	<code>\nvarleq</code>
\gtrapprox	<code>\gtrapprox</code>	\lesseqqgtr	<code>\lesseqqgtr</code>	\ngeq	<code>\vargeq</code>
\gtrdot	<code>\gtrdot</code>	\lessgtr	<code>\lessgtr</code>	\ngeqq	<code>\varleq</code>

表 47 AMS 三角形关系符号

\blacktriangleleft	<code>\blacktriangleleft</code>	\ntriangleright	<code>\ntriangleright</code>	\trianglerighteq	<code>\trianglerighteq</code>
\blacktriangleright	<code>\blacktriangleright</code>	\ntrianglerighteq	<code>\ntrianglerighteq</code>	\vartriangleleft	<code>\vartriangleleft</code>
\ntriangleleft	<code>\ntriangleleft</code>	\trianglelefteq	<code>\trianglelefteq</code>	\vartriangleright	<code>\vartriangleright</code>
\ntrianglelefteq	<code>\ntrianglelefteq</code>	\trianglelefteq	<code>\trianglelefteq</code>		

表 48 stmaryrd 三角形关系符号

\trianglelefteqslant	<code>\trianglelefteqslant</code>	\trianglerighteqslant	<code>\trianglerighteqslant</code>
\ntrianglelefteqslant	<code>\ntrianglelefteqslant</code>	\ntrianglerighteqslant	<code>\ntrianglerighteqslant</code>

表 49 mathabx 三角形关系符号

\triangleleft	<code>\ntriangleleft</code>	\triangleleft	<code>\triangleleft</code>	\triangleleft	<code>\vartriangleleft</code>
\trianglelefteq	<code>\ntrianglelefteq</code>	\trianglelefteq	<code>\trianglelefteq</code>	\triangleright	<code>\vartriangleright</code>
\triangleright	<code>\ntriangleright</code>	\triangleright	<code>\triangleright</code>		
\trianglerighteq	<code>\ntrianglerighteq</code>	\trianglerighteq	<code>\trianglerighteq</code>		

表 50 箭头符号

\Downarrow	<code>\Downarrow</code>	\longleftarrow	<code>\longleftarrow</code>	\Rightarrow	<code>\Rightarrow</code>
\downarrow	<code>\downarrow</code>	\Longleftarrow	<code>\Longleftarrow</code>	\rightarrow	<code>\rightarrow</code>
\hookleftarrow	<code>\hookleftarrow</code>	\longleftrightarrow	<code>\longleftrightarrow</code>	\rightarrow	<code>\rightarrow</code>
\hookrightarrow	<code>\hookrightarrow</code>	\Longleftrightarrow	<code>\Longleftrightarrow</code>	\Rightarrow	<code>\Rightarrow</code>
\leadsto	<code>\leadsto</code>	\mapsto	<code>\mapsto</code>	\searrow	<code>\searrow</code>
\Leftarrow	<code>\Leftarrow</code>	\longrightarrow	<code>\longrightarrow</code>	\swarrow	<code>\swarrow</code>
\leftarrow	<code>\leftarrow</code>	\Longrightarrow	<code>\Longrightarrow</code>	\Uparrow	<code>\Uparrow</code>
\leftharpoonup	<code>\leftharpoonup</code>	\mapsto	<code>\mapsto</code>	\Uparrow	<code>\Uparrow</code>
\leftharpoonup	<code>\leftharpoonup</code>	\nearrow	<code>\nearrow</code>	\Updownarrow	<code>\Updownarrow</code>
\Leftrightarrow	<code>\Leftrightarrow</code>	\nwarrow	<code>\nwarrow</code>	\updownarrow	<code>\updownarrow</code>
\leftrightarrow	<code>\leftrightarrow</code>	\rightarrow	<code>\rightarrow</code>		

* 在 L^AT_EX 内没有定义, 必须调入以下宏包之一: latexsym, amsfonts, amssymb, txfonts, pxfonts, wasysym.

表 51 AMS 箭头符号

\circlearrowleft	<code>\circlearrowleft</code>	\leftrightsquigarrow	<code>\leftrightsquigarrow</code>	\rightleftarrows	<code>\rightleftarrows</code>
\circlearrowright	<code>\circlearrowright</code>	\leftrightsquigarrow	<code>\leftrightsquigarrow</code>	\rightleftharpoons	<code>\rightleftharpoons</code>
\curvearrowleft	<code>\curvearrowleft</code>	\leftrightsquigarrow	<code>\leftrightsquigarrow</code>	\rightrightarrows	<code>\rightrightarrows</code>
\curvearrowright	<code>\curvearrowright</code>	\leftrightsquigarrow	<code>\leftrightsquigarrow</code>	\rightsquigarrow	<code>\rightsquigarrow</code>
\dashleftarrow	<code>\dashleftarrow</code>	\Lleftarrow	<code>\Lleftarrow</code>	\Rsh	<code>\Rsh</code>
\dashrightarrow	<code>\dashrightarrow</code>	\looparrowleft	<code>\looparrowleft</code>	\twoheadleftarrow	<code>\twoheadleftarrow</code>
\downdownarrows	<code>\downdownarrows</code>	\looparrowright	<code>\looparrowright</code>	\twoheadrightarrow	<code>\twoheadrightarrow</code>
\downharpoonleft	<code>\downharpoonleft</code>	\Lsh	<code>\Lsh</code>	\upharpoonleft	<code>\upharpoonleft</code>
\downharpoonright	<code>\downharpoonright</code>	\multimap	<code>\multimap</code>	\upharpoonright	<code>\upharpoonright</code>
\leftarrowtail	<code>\leftarrowtail</code>	\rightarrowtail	<code>\rightarrowtail</code>	\upuparrows	<code>\upuparrows</code>

表 52 AMS 否定箭头符号

\nLeftarrow	<code>\nLeftarrow</code>	\nLeftrightarrow	<code>\nLeftrightarrow</code>	\nRightarrow	<code>\nRightarrow</code>
\nleftarrow	<code>\nleftarrow</code>	\nleftrightsquigarrow	<code>\nleftrightsquigarrow</code>	\nrightarrow	<code>\nrightarrow</code>

表 53 stmaryrd 箭头符号

\leftarrow	<code>\leftarrowtriangle</code>	\Leftarrow	<code>\Mapsfrom</code>	\leftarrow	<code>\shortleftarrow</code>
\Leftrightarrow	<code>\leftrightharroweq</code>	\mapsto	<code>\mapsfrom</code>	\rightarrow	<code>\shortrightarrow</code>
\Leftrightarrow	<code>\leftrightharrowtriangle</code>	\mapsto	<code>\Mapsto</code>	\uparrow	<code>\shortuparrow</code>
\lightning	<code>\lightning</code>	\nearrow	<code>\nnearrow</code>	\searrow	<code>\ssearrow</code>
\Longmapsfrom	<code>\Longmapsfrom</code>	\nwarrow	<code>\nnwarrow</code>	\swarrow	<code>\sswarrow</code>
\longmapsfrom	<code>\longmapsfrom</code>	\rightarrowtriangle	<code>\rightarrowtriangle</code>		
\Longmapsto	<code>\Longmapsto</code>	\shortdownarrow	<code>\shortdownarrow</code>		

表 54 txfonts/pxfonts 箭头符号

\boxdotleft	<code>\boxdotLeft</code>	\circrightarrow	<code>\circleddotright</code>	\diamondleft	<code>\Diamondleft</code>
\boxdotleft	<code>\boxdotleft</code>	\circleft	<code>\circleleft</code>	\diamondrightarrow	<code>\Diamondright</code>
\boxdotright	<code>\boxdotright</code>	\circrightarrow	<code>\circcleright</code>	\DiamondRight	<code>\DiamondRight</code>
\boxdotright	<code>\boxdotRight</code>	\dashleftarrow	<code>\dashleftarrow</code>	\leftsquigarrow	<code>\leftsquigarrow</code>
\boxleft	<code>\boxLeft</code>	$\Diamond\dotleft$	<code>\DiamondddotLeft</code>	\nearrow	<code>\Nearrow</code>
\boxleft	<code>\boxleft</code>	$\Diamond\dotleft$	<code>\Diamondddotleft</code>	\nwarrow	<code>\Nwarrow</code>
\boxright	<code>\boxright</code>	$\Diamond\dotright$	<code>\Diamondddotright</code>	\Rightarrow	<code>\Rrightarrow</code>
\boxright	<code>\boxRight</code>	$\Diamond\dotright$	<code>\DiamondddotRight</code>	\searrow	<code>\Searrow</code>
\circleft	<code>\circleddotleft</code>	\DiamondLeft	<code>\DiamondLeft</code>	\swarrow	<code>\Swarrow</code>

表 55 mathabx 箭头符号

\circlearrowleft	<code>\circlearrowleft</code>	\leftarrow	<code>\leftarrow</code>	\nwarrow	<code>\nwarrow</code>
\circlearrowright	<code>\circlearrowright</code>	\leftleftarrows	<code>\leftleftarrows</code>	\restriction	<code>\restriction</code>
\curvearrowleft	<code>\curvearrowleft</code>	\leftrightarrow	<code>\leftrightarrow</code>	\rightarrow	<code>\rightarrow</code>
\curvearrowleft	<code>\curvearrowleft</code>	\leftrightarrows	<code>\leftrightarrows</code>	\rightleftarrows	<code>\rightleftarrows</code>
\curvearrowright	<code>\curvearrowright</code>	\leftrightsquigarrow	<code>\leftrightsquigarrow</code>	\rightrightarrows	<code>\rightrightarrows</code>
\curvearrowleft	<code>\curvearrowleft</code>	\leftsquigarrow	<code>\leftsquigarrow</code>	\rightsquigarrow	<code>\rightsquigarrow</code>
\curvearrowleft	<code>\curvearrowleft</code>	\lefttorightarrow	<code>\lefttorightarrow</code>	\righttoleftarrow	<code>\righttoleftarrow</code>
\curvearrowright	<code>\curvearrowright</code>	\looparrowleft	<code>\looparrowleft</code>	\Rsh	<code>\Rsh</code>
\dls	<code>\dls</code>	\looparrowright	<code>\looparrowright</code>	\searrow	<code>\searrow</code>
\downarrow	<code>\downarrow</code>	\looparrowleft	<code>\looparrowleft</code>	\swarrow	<code>\swarrow</code>
\downtoup	<code>\downtoup</code>	\looparrowright	<code>\looparrowright</code>	\updownarrow	<code>\updownarrow</code>
\downuparrow	<code>\downuparrow</code>	\Lsh	<code>\Lsh</code>	\uptodownarrow	<code>\uptodownarrow</code>
\drsh	<code>\drsh</code>	\nearrow	<code>\nearrow</code>	\upuparrows	<code>\upuparrows</code>

表 56 mathabx 否定箭头符号

\nleftarrow	<code>\nleftarrow</code>	\nleftrightarrow	<code>\nleftrightarrow</code>	\nrightarrow	<code>\nrightarrow</code>
\nleftarrow	<code>\nleftarrow</code>	\nLefttrightarrow	<code>\nLefttrightarrow</code>	\nRightarrow	<code>\nRightarrow</code>

表 57 mathabx 单侧箭头符号

\Rightarrow	<code>\barleftharpoon</code>	\leftarrow	<code>\leftharpoonup</code>	\Rightarrow	<code>\rightleftharpoons</code>
\Rightarrow	<code>\barrightharpoon</code>	\Leftarrow	<code>\leftleftharpoons</code>	\Rightarrow	<code>\rightrightarpoons</code>
\Downarrow	<code>\downdownharpoons</code>	\leftrightarrow	<code>\leftrightharpoon</code>	\Updownarrow	<code>\updownharpoons</code>
\downarrow	<code>\downharpoonleft</code>	\Leftrightarrow	<code>\leftrightharpoons</code>	\uparrow	<code>\upharpoonleft</code>
\downarrow	<code>\downharpoonright</code>	\Rightarrow	<code>\rightbarharpoon</code>	\uparrow	<code>\upharpoonright</code>
\Downarrow	<code>\downupharpoons</code>	\rightarrow	<code>\rightharpoondown</code>	\Uparrow	<code>\upupharpoons</code>
\Leftarrow	<code>\leftbarharpoon</code>	\rightarrow	<code>\rightharpoonup</code>		
\Leftarrow	<code>\leftharpoondown</code>	\rightarrow	<code>\rightleftharpoon</code>		

表 58 ulsy 矛盾符号

⚡	<code>\blitza</code>	⚡	<code>\blitzb</code>	⚡	<code>\blitzc</code>	⚡	<code>\blitzd</code>	⚡	<code>\blitze</code>
------------	----------------------	------------	----------------------	------------	----------------------	------------	----------------------	------------	----------------------

表 59 \mathcal{AMS} 希腊字母

\mathcal{F}	<code>\digamma</code>	\mathcal{K}	<code>\varkappa</code>
---------------	-----------------------	---------------	------------------------

表 60 txfonts/pxfonts 直立体希腊字母

α	<code>\alphaup</code>	θ	<code>\thetaup</code>	π	<code>\piup</code>	ϕ	<code>\phiup</code>
β	<code>\betaup</code>	ϑ	<code>\varthetaup</code>	ϖ	<code>\varpiup</code>	φ	<code>\varphiup</code>
γ	<code>\gammaup</code>	ι	<code>\iotaup</code>	ρ	<code>\rhoup</code>	χ	<code>\chiup</code>
δ	<code>\deltaup</code>	κ	<code>\kappaup</code>	ϱ	<code>\varrhoup</code>	ψ	<code>\psiup</code>
ϵ	<code>\epsilonup</code>	λ	<code>\lambdaup</code>	σ	<code>\sigmaup</code>	ω	<code>\omegaup</code>
ε	<code>\varepsilonup</code>	μ	<code>\muup</code>	ς	<code>\varsigmaup</code>		
ζ	<code>\zetaup</code>	ν	<code>\nuup</code>	τ	<code>\tauup</code>		
η	<code>\etaup</code>	ξ	<code>\xiup</code>	υ	<code>\upsilonup</code>		

表 61 upgreek 直立体希腊字母

α	<code>\upalpha</code>	θ	<code>\uptheta</code>	π	<code>\uppi</code>	ϕ	<code>\upphi</code>
β	<code>\upbeta</code>	ϑ	<code>\upvartheta</code>	ϖ	<code>\upvarpi</code>	φ	<code>\upvarphi</code>
γ	<code>\upgamma</code>	ι	<code>\upiota</code>	ρ	<code>\uprho</code>	χ	<code>\upchi</code>
δ	<code>\updelta</code>	κ	<code>\upkappa</code>	ϱ	<code>\upvarrho</code>	ψ	<code>\uppsi</code>
ϵ	<code>\upepsilon</code>	λ	<code>\uplambda</code>	σ	<code>\upsigma</code>	ω	<code>\upomega</code>
ε	<code>\upvarepsilon</code>	μ	<code>\upmu</code>	ς	<code>\upvarsigma</code>		
ζ	<code>\upzeta</code>	ν	<code>\upnu</code>	τ	<code>\uptau</code>		
η	<code>\upeta</code>	ξ	<code>\upxi</code>	υ	<code>\upupsilon</code>		
Γ	<code>\Upgamma</code>	Λ	<code>\Uplambda</code>	Σ	<code>\Upsilonsigma</code>	Ψ	<code>\Uppsi</code>
Δ	<code>\Updelta</code>	Ξ	<code>\Upxi</code>	Υ	<code>\Upupsilon</code>	Ω	<code>\Upomega</code>
Θ	<code>\Uptheta</code>	Π	<code>\Uppi</code>	Φ	<code>\Upphi</code>		

表 62 txfonts/pxfonts 变体拉丁字母

g `\varg` v `\varv` w `\varw` y `\vary`

若在调入 txfonts/pxfonts 时加入选项 varg, 可把所有数学公式里的 g, v, w, y 换成变体.

表 63 AMS 希伯来字母

\beth `\beth` \daleth `\daleth` \gimel `\gimel`

\aleph 在 L^AT_EX 中已有定义.

表 64 AMS 定界符

\ulcorner `\ulcorner` \urcorner `\urcorner` \llcorner `\llcorner` \lrcorner `\lrcorner`

表 65 stmaryrd 定界符

\Lbag `\Lbag` \Rbag `\Rbag` \lbag `\lbag` \rbag `\rbag`
 \llceil `\llceil` \rrceil `\rrceil` \llfloor `\llfloor` \rrfloor `\rrfloor`
 \llparenthesis `\llparenthesis` \rrparenthesis `\rrparenthesis`

表 66 mathabx 定界符

\lcorners `\lcorners` \ulcorner `\ulcorner` \urcorner `\urcorner`
 \rcorners `\rcorners` \llcorner `\llcorner` \lrcorner `\lrcorner`

表 67 nath 定界符

\niv `\niv` \vin `\vin`

表 68 可变大小的定界符

\downarrow `\downarrow` \Downarrow `\Downarrow` $[$ `[` $]$ `]`
 \langle `\langle` \rangle `\rangle \langle \langle \rangle \rangle \lvert \lvert \rvert \rvert
 \lceil \lceil \rceil \rceil \uparrow \uparrow \Uparrow \Uparrow
 \lfloor \lfloor \rfloor \rfloor \updownarrow \updownarrow \Updownarrow \Updownarrow
 $($ ($)$) $\{$ \{ $\}$ \}
 $/$ / \backslash \backslash`

这里以及后面的可变大小的定界符应该与 `\left` 及 `\right` 合用.

表 69 可变大小的定界符

$$\int \int \backslash \text{lmoustache} \left(\right) \backslash \text{rmoustache} \left(\left(\backslash \text{lgroup} \right) \right) \backslash \text{rgroup}$$

$$\mid \mid \backslash \text{arrowvert} \parallel \parallel \backslash \text{Arrowvert} \mid \mid \backslash \text{bracevert}$$
表 70 *AMS* 可变大小的定界符
$$\mid \mid \backslash \text{lvert} \mid \mid \backslash \text{rvert} \parallel \parallel \backslash \text{lVert} \parallel \parallel \backslash \text{rVert}$$
表 71 可变大小的 *stmaryrd* 定界符
$$\llbracket \llbracket \backslash \text{llbracket} \rrbracket \rrbracket \backslash \text{rrbracket}$$
表 72 可变大小的 *mathabx* 定界符
$$\llbracket \llbracket \backslash \text{lbbbrack} \rrbracket \rrbracket \backslash \text{rbbbrack} \mid \mid \backslash \text{thickvert} \parallel \parallel \backslash \text{vvvert}$$

$$\left. \left. \backslash \text{lfilet} \right. \right. \left. \left. \backslash \text{rfilet} \right. \right.$$
表 73 可变大小的 *nath* 定界符 (二重)
$$\ll \ll \backslash \text{lAngle} \gg \gg \backslash \text{rAngle} \ll \ll \backslash \text{lFloor} \gg \gg \backslash \text{rFloor}$$

$$\llbracket \llbracket \backslash \text{lBrack} \rrbracket \rrbracket \backslash \text{rBrack} \parallel \parallel \backslash \text{lVert} \parallel \parallel \backslash \text{rVert}$$

$$\lceil \lceil \backslash \text{lCeil} \rceil \rceil \backslash \text{rCeil}$$

由于表 73 与 74 中 *nath* 命令已经蕴含了 `\left` 及 `\right`, 因此使用时不需再添加两个命令.

表 74 可变大小的 *nath* 定界符 (三重)
$$\lll \lll \backslash \text{triple} < \ggg \ggg \backslash \text{triple} > \lll \lll \backslash \text{triple} | \lll \lll \backslash \text{triple} |$$

$$\lll \lll \backslash \text{triple} [\lll \lll \backslash \text{triple}]$$

表 75 数学模式重音号

\acute{a}	<code>\acute{a}</code>	\check{a}	<code>\check{a}</code>	\grave{a}	<code>\grave{a}</code>	\tilde{a}	<code>\tilde{a}</code>
\bar{a}	<code>\bar{a}</code>	\ddot{a}	<code>\ddot{a}</code>	\hat{a}	<code>\hat{a}</code>	\vec{a}	<code>\vec{a}</code>
\breve{a}	<code>\breve{a}</code>	\dot{a}	<code>\dot{a}</code>	\mathring{a}	<code>\mathring{a}</code>		

表 76 AMS 的数学模式重音号

\dddot{a}	<code>\dddot{a}</code>	\ddddot{a}	<code>\ddddot{a}</code>
-------------	------------------------	--------------	-------------------------

表 77 可延伸的置顶或置底符号

\widetilde{abc}	<code>\widetilde{abc}</code> *	\widehat{abc}	<code>\widehat{abc}</code> *
\overleftarrow{abc}	<code>\overleftarrow{abc}</code>	\overrightarrow{abc}	<code>\overrightarrow{abc}</code>
\overline{abc}	<code>\overline{abc}</code>	\underline{abc}	<code>\underline{abc}</code>
\overbrace{abc}	<code>\overbrace{abc}</code>	\underbrace{abc}	<code>\underbrace{abc}</code>
\sqrt{abc}	<code>\sqrt{abc}</code>		

* 在 yhm_{ath} 包里有更大的伸展性.

表 78 yhm_{ath} 的可延伸置顶或置底符号

\wideparen{abc}	<code>\wideparen{abc}</code>	$\widehat{\triangle} abc$	<code>\widehat{\triangle} abc</code>
$\widetilde{\circ} abc$	<code>\widetilde{\circ} abc</code>		

表 79 AMS 的可延伸置顶或置底符号

\overleftrightarrow{abc}	<code>\overleftrightarrow{abc}</code>	\underleftarrow{abc}	<code>\underleftarrow{abc}</code>
\underlineleftarrow{abc}	<code>\underlineleftarrow{abc}</code>	\underrightarrow{abc}	<code>\underrightarrow{abc}</code>

表 80 emp_{heq} 的可延伸置顶或置底符号

\overbracket{abc}	<code>\overbracket{abc}</code>	\underbracket{abc}	<code>\underbracket{abc}</code>
---------------------	--------------------------------	----------------------	---------------------------------

表 81 undertilde 的可延伸置顶或置底符号

\utilde{abc}	<code>\utilde{abc}</code>
----------------	---------------------------

表 82 mathabx 的可延伸置顶或置底符号

\overbrace{abc}	<code>\overbrace{abc}</code>	\overline{abc}	<code>\widebar{abc}</code>
\overgroup{abc}	<code>\overgroup{abc}</code>	\widetilde{abc}	<code>\widecheck{abc}</code>
\underbrace{abc}	<code>\underbrace{abc}</code>	\underbar{abc}	<code>\wideparen{abc}</code>
\undergroup{abc}	<code>\undergroup{abc}</code>	$\underaccent{\circ}{abc}$	<code>\widering{abc}</code>
\overrightarrow{abc}	<code>\widearrow{abc}</code>		

这里显示了 `\overbrace` 与 `\underbrace` 命令的最小宽度, 而最大宽度并无限制.

表 83 AMS 可延伸的加标注箭头符号

\xleftarrow{abc}	<code>\xleftarrow{abc}</code>	\xrightarrow{abc}	<code>\xrightarrow{abc}</code>
--------------------	-------------------------------	---------------------	--------------------------------

表 84 empheq 的可延伸的加标注箭头符号

\hookleftarrow{abc}	<code>\xhookleftarrow{abc}</code>	\rightharpoonleft{abc}	<code>\xleftrightharpoons{abc}</code>
\hookrightarrow{abc}	<code>\xhookrightarrow{abc}</code>	\mapsto{abc}	<code>\xmapsto{abc}</code>
\Lleftarrow{abc}	<code>\xLeftarrow{abc}</code>	\Rrightarrow{abc}	<code>\xRrightarrow{abc}</code>
\leftharpoonup{abc}	<code>\xleftharpoonupdown{abc}</code>	\rightharpoonup{abc}	<code>\xrightarpoonupdown{abc}</code>
\leftharpoonup{abc}	<code>\xleftharpoonup{abc}</code>	\rightharpoonup{abc}	<code>\xrightarpoonup{abc}</code>
\Lleftrightarrow{abc}	<code>\xLeftrightarrow{abc}</code>	\Rleftrightarrow{abc}	<code>\xrightleftharpoons{abc}</code>
\longleftrightarrow{abc}	<code>\xleftrightarrow{abc}</code>		

表 85 extarrows 的可延伸的加标注箭头符号

\Lleftrightarrow{abc}	<code>\xLeftrightarrow{abc}</code>	\Longleftrightarrow{abc}	<code>\xLongleftrightarrow{abc}</code>
\xleftrightarrow{abc}	<code>\xleftrightarrow{abc}</code>	$\xlongleftrightarrow{abc}$	<code>\xlongleftrightarrow{abc}</code>
\xlongequal{abc}	<code>\xlongequal{abc}</code>	\xrightarrow{abc}	<code>\xrightarrow{abc}</code>
\xrightarrow{abc}	<code>\xrightarrow{abc}</code>	\xrightarrow{abc}	<code>\xrightarrow{abc}</code>
\xrightarrow{abc}	<code>\xrightarrow{abc}</code>		

表 86 holtpolt 的非交换除法符号

$\frac{abc}{def}$	<code>\holter{abc}{def}</code>	$\frac{abc}{def}$	<code>\polter{abc}{def}</code>
-------------------	--------------------------------	-------------------	--------------------------------

表 87 点列符号

\cdot	<code>\cdotp</code>	:	<code>\colon</code>	\cdot	<code>\ldotp</code>	\vdots	<code>\vdots*</code>
\cdots	<code>\cdots</code>	\ddots	<code>\ddots*</code>	\dots	<code>\ldots</code>		

* `mathdots` 包重新定义 `\ddots` 与 `\vdots`, 使它们与周围字符的大小匹配.

表 88 *AMS* 的点列符号

\cdots	<code>\dotsb</code>	\cdots	<code>\dotsi</code>	\cdots	<code>\dotso</code>
\cdots	<code>\dotsc</code>	\cdots	<code>\dotsm</code>		

表 89 `mathdots` 的点列符号

\cdots	<code>\iddots</code>
----------	----------------------

表 90 `yhmath` 的点列符号

\cdots	<code>\adots</code>
----------	---------------------

表 91 `mathcomp` 的数学符号

$^{\circ}\text{C}$	<code>\tccentigrade</code>	Ω	<code>\tcohm</code>	$\%$	<code>\tcpertousand</code>
μ	<code>\tcmu</code>	$\%$	<code>\tcpertenthousand</code>		

表 92 `mathabx` 的玛雅数字

$\textcircled{0}$	<code>\maya{0}</code>	:	<code>\maya{2}</code>	:	<code>\maya{4}</code>
\cdot	<code>\maya{1}</code>	:	<code>\maya{3}</code>	:	<code>\maya{5}</code>

表 93 `marvosym` 的数学符号

0	<code>\MVZero</code>	2	<code>\MVTwo</code>	4	<code>\MVFour</code>	6	<code>\MVSix</code>	8	<code>\MVEight</code>
1	<code>\MVOne</code>	3	<code>\MVThree</code>	5	<code>\MVFive</code>	7	<code>\MVSeven</code>	9	<code>\MVNine</code>
\angle	<code>\Anglesign</code>	\cdot	<code>\Squaredot</code>	\rightarrow	<code>\Vectorarrowhigh</code>				
\cong	<code>\Corresponds</code>	\rightarrow	<code>\Vectorarrow</code>						

表 94 其他 L^AT_EX 符号

\aleph	<code>\aleph</code>	\diamond	<code>\Diamond*</code>	∞	<code>\infty</code>	$'$	<code>\prime</code>
\angle	<code>\angle</code>	\diamond	<code>\diamondsuit</code>	\cup	<code>\mho*</code>	\sharp	<code>\sharp</code>
\backslash	<code>\backslash</code>	\emptyset	<code>\emptyset†</code>	∇	<code>\nabla</code>	\spadesuit	<code>\spadesuit</code>
\Box	<code>\Box*,†</code>	\flat	<code>\flat</code>	\natural	<code>\natural</code>	\surd	<code>\surd</code>
\clubsuit	<code>\clubsuit</code>	\heartsuit	<code>\heartsuit</code>	\neg	<code>\neg</code>	\triangle	<code>\triangle</code>

* 在 L^AT_EX 内没有定义, 必须调入以下宏包之一: `latexsym`, `amsfonts`, `amssymb`, `txfonts`, `pxfonts`, `wasysm`.

† 如果使用 `\Box` 作为证毕符号, 建议使用 `ntheorem` 包, 可以安放在合适的位置.

‡ 许多人喜欢用 *AMS* 的 `\varnothing`.

表 95 其他 AMS 符号

\angle	<code>\angle</code>	\blacktriangledown	<code>\blacktriangledown</code>	\mho	<code>\mho</code>
\backprime	<code>\backprime</code>	\diagdown	<code>\diagdown</code>	\sphericalangle	<code>\sphericalangle</code>
\bigstar	<code>\bigstar</code>	\diagup	<code>\diagup</code>	\square	<code>\square</code>
\blacklozenge	<code>\blacklozenge</code>	\eth	<code>\eth</code>	\triangledown	<code>\triangledown</code>
\blacksquare	<code>\blacksquare</code>	\lozenge	<code>\lozenge</code>	\varnothing	<code>\varnothing</code>
\blacktriangle	<code>\blacktriangle</code>	\measuredangle	<code>\measuredangle</code>	\vartriangle	<code>\vartriangle</code>

表 96 数学与文本模式都能使用的 AMS 命令

\checkmark	<code>\checkmark</code>	\textcircled{R}	<code>\circledR</code>	✂	<code>\maltese</code>
--------------	-------------------------	-------------------	------------------------	------------	-----------------------

表 97 wasysym 的其他数学符号

\Box	<code>\Box</code>	\mho^*	<code>\mho^*</code>	\therefore	<code>\wasytherefore</code>
\Diamond	<code>\Diamond</code>	\varangle	<code>\varangle</code>		

表 98 txfonts/pxfonts 的其他数学符号

\blacklozenge	<code>\Diamondblack</code>	λ	<code>\lambdaslash</code>	\heartsuit	<code>\varheartsuit</code>
\diamond	<code>\Diamonddot</code>	\clubsuit	<code>\varclubsuit</code>	\spadesuit	<code>\varspadesuit</code>
λ	<code>\lambdabar</code>	\blacklozenge	<code>\vardiamondsuit</code>		

表 99 mathabx 的其他数学符号

\circ	<code>\degree</code>	///	<code>\fourth</code>	\measuredangle	<code>\measuredangle</code>	''	<code>\second</code>
\diagdown	<code>\diagdown</code>	$\#$	<code>\hash</code>	\pitchfork	<code>\pitchfork</code>	\sphericalangle	<code>\sphericalangle</code>
\diagup	<code>\diagup</code>	∞	<code>\infty</code>	\propto	<code>\propto</code>	///	<code>\third</code>
\oslash	<code>\diameter</code>	\times	<code>\leftthreetimes</code>	\times	<code>\rightthreetimes</code>	$\#$	<code>\varhash</code>

表 100 数学字母表

		需用宏包
$\mathrm{ABCdef123}$	<code>\mathrm{ABCdef123}</code>	无
$\mathit{ABCdef123}$	<code>\mathit{ABCdef123}</code>	无
$\mathnormal{ABCdef123}$	<code>\mathnormal{ABCdef123}</code>	无
\mathcal{ABC}	<code>\mathcal{ABC}</code>	无
\mathscr{ABC}	<code>\mathscr{ABC}</code>	无
	<code>\mathrsfs</code>	
	或 <code>\mathcal{ABC}</code>	<code>\calrsfs</code>
\mathcal{ABC}	<code>\mathcal{ABC}</code>	euscript 带选项: <code>\mathcal</code>
	或 <code>\mathscr{ABC}</code>	euscript 带选项: <code>\mathscr</code>

<i>ABCdef123</i>	<code>\mathpzc{ABCdef123}</code>	无; 手工定义*
ABC	<code>\mathbb{ABC}</code>	amssymb, txfonts 或 pxfonts
ABC	<code>\varmathbb{ABC}</code>	txfonts 或 pxfonts
ABCdef123	<code>\mathbb{ABCdef123}</code>	bbold 或 mathbbol [†]
ABCdef12	<code>\mathbbm{ABCdef12}</code>	bbm
ABCdef12	<code>\mathbbmss{ABCdef12}</code>	bbm
ABCdef12	<code>\mathbbmtt{ABCdef12}</code>	bbm
ABC1	<code>\mathds{ABC1}</code>	dsfont
A\BC1	<code>\mathds{ABC1}</code>	dsfont 带选项: sans
<i>ABCdef123</i>	<code>\mathfrak{ABCdef123}</code>	eufrak
<i>ABCdef123</i>	<code>\textfrak{ABCdef123}</code>	yfonts
<i>ABCdef123</i>	<code>\textswab{ABCdef123}</code>	yfonts
<i>ABCdef123</i>	<code>\textgoth{ABCdef123}</code>	yfonts

* 为了使 `\mathpzc` 能打印出 Zapf Chancery 字体, 必须在导言里输入:
`\DeclareMathAlphabet{\mathpzc}{OT1}{pzc}{m}{it}`.
[†] `mathbbol` 包定义了更多的黑板黑体: 圆括号, 方括号, 尖括号等. 如果调入 `mathbbol` 宏包时加入了选项 `bbgreek1`, 那么希腊字母也有黑板黑体.

第三部分 科技符号









表 101 gensymb 的数学与文本模式共用的符号

°C	<code>\celsius</code>	μ	<code>\micro</code>	‰	<code>\perthousand</code>
°	<code>\degree</code>	Ω	<code>\ohm</code>		

表 102 wasysym 的物理符号

~	<code>\AC</code>	≈	<code>\VHF</code>	~~~~	<code>\photon</code>	≈	<code>\HF</code>	⊗⊗⊗⊗	<code>\gluon</code>
---	------------------	---	-------------------	------	----------------------	---	------------------	------	---------------------

表 103 ifsym 脉冲图形符号

	<code>\FallingEdge</code>		<code>\PulseHigh</code>		<code>\ShortPulseHigh</code>
	<code>\LongPulseHigh</code>		<code>\PulseLow</code>		<code>\ShortPulseLow</code>
	<code>\LongPulseLow</code>		<code>\RaisingEdge</code>		

为使用本组符号, 必须在调入 ifsym 时使用选项 `electronic`.

表 104 ar 的纵横比符号

\mathcal{R}	<code>\AR</code>
---------------	------------------

表 105 wasysym 的天文符号

Ω	<code>\ascnode</code>	$\♃$	<code>\jupiter</code>	\bullet	<code>\newmoon</code>	$\♀$	<code>\venus</code>
\odot	<code>\astrosun</code>	$\☾$	<code>\leftmoon</code>	$\♇$	<code>\pluto</code>	$\♈$	<code>\vernal</code>
\oslash	<code>\descnode</code>	$\♂$	<code>\mars</code>	$\☾$	<code>\rightmoon</code>		
$\♁$	<code>\earth</code>	$\♂$	<code>\mercury</code>	$\♄$	<code>\saturn</code>		
\bigcirc	<code>\fullmoon</code>	$\♆$	<code>\neptune</code>	$\♅$	<code>\uranus</code>		

表 106 marvosym 的天文符号

$\♁$	<code>\Mercury</code>	$\♂$	<code>\Mars</code>	$\♆$	<code>\Uranus</code>	\odot	<code>\Sun</code>
$\♀$	<code>\Venus</code>	$\♃$	<code>\Jupiter</code>	$\♆$	<code>\Neptune</code>	$\☾$	<code>\Moon</code>
$\♁$	<code>\Earth</code>	$\♄$	<code>\Saturn</code>	$\♇$	<code>\Pluto</code>		

表 107 mathabx 的天文符号

$\♁$	<code>\Mercury</code>	\oplus	<code>\Earth</code>	$\♃$	<code>\Jupiter</code>	$\♅$	<code>\Uranus</code>	$\♇$	<code>\Pluto</code>
$\♀$	<code>\Venus</code>	$\♂$	<code>\Mars</code>	$\♄$	<code>\Saturn</code>	$\♆$	<code>\Neptune</code>		
\bigcirc	<code>\fullmoon</code>	$\☾$	<code>\leftmoon</code>	\bullet	<code>\newmoon</code>	$\☾$	<code>\rightmoon</code>		
\odot	<code>\Sun</code>	\oplus	<code>\varEarth</code>						

表 108 wasysym 的星相符号

$\♈$	<code>\aries</code>	$\♋$	<code>\cancer</code>	$\♎$	<code>\libra</code>	$\♏$	<code>\capricornus</code>
$\♉$	<code>\taurus</code>	$\♌$	<code>\leo</code>	$\♏$	<code>\scorpio</code>	$\♐$	<code>\aquarius</code>
$\♊$	<code>\gemini</code>	$\♍$	<code>\virgo</code>	$\♐$	<code>\sagittarius</code>	$\♑$	<code>\pisces</code>
		$\♈$	<code>\conjunction</code>	$\♈$	<code>\opposition</code>		

表 109 marvosym 的星相符号

$\♈$	<code>\Aries</code>	$\♋$	<code>\Cancer</code>	$\♎$	<code>\Libra</code>	$\♏$	<code>\Capricorn</code>
$\♉$	<code>\Taurus</code>	$\♌$	<code>\Leo</code>	$\♏$	<code>\Scorpio</code>	$\♐$	<code>\Aquarius</code>
$\♊$	<code>\Gemini</code>	$\♍$	<code>\Virgo</code>	$\♐$	<code>\Sagittarius</code>	$\♑$	<code>\Pisces</code>

也可用 `\Zodiac{1}...``\Zodiac{12}` 代替 `\Aries...``\Pisces`.

表 110 mathabx 的星相符号

$\♈$	<code>\Aries</code>	$\♉$	<code>\Taurus</code>	$\♊$	<code>\Gemini</code>
------	---------------------	------	----------------------	------	----------------------

表 111 wasysym 的 APL 符号

	\APLbox		\APLinv	*	\APLstar
	\APLcomment		\APLleftarrowbox	Δ	\APLup
∇	\APLdown	\otimes	\APLlog		\APLuparrowbox
	\APLdownarrowbox	-	\APLminus	\nmid	\notbackslash
	\APLinput		\APLrightarrowbox	\nmid	\notslash
◦ \APLcirc{} ~ \APLnot{} \APLvert{}					

表 112 marvosym 的计算机硬件符号

	\ComputerMouse		\ParallelPort		\SerialInterface
	\Keyboard		\Printer		\SerialPort

表 113 ASCII 控制符 (IBM)

☺	\SOH	•	\BEL	♪	\CR	!!	\DCc	↓	\EM	▼	\US
☹	\STX	■	\BS	♫	\SO	¶	\DCd	→	\SUB		\splitvert
♥	\ETX	○	\HT	✱	\SI	\$	\NAK	←	\ESC	△	\DEL
♦	\EOT	◼	\LF	▶	\DLE	-	\SYN	L	\FS		
♣	\ENQ	♂	\VT	◀	\DCa	‡	\ETB	↔	\GS		
♠	\ACK	♀	\FF	‡	\DCb	↑	\CAN	▲	\RS		

输出这些字符需要 `ascii` 宏包以及 `ascii` 字体, 例如: `{\ascii\STX}`. 详见此宏包的说明文件.

表 114 marvosym 的通讯符号

	\Email		\fax		\Faxmachine	⚡	\Lightning	○	\Pickup
⚡	\Emailct		\FAX		\Letter		\Mobilefone		\Telefon

表 115 marvosym 的工程符号

	\Beam	↓	\Force	●	\Octosteel	I	\RoundedTTsteel
	\Bearing	●	\Hexasteel	□	\Rectpipe	□	\Squarepipe
○	\Circpipe	↺	\Lefttorque	■	\Rectsteel	■	\Squaresteel
●	\Circsteel	≡	\Lineload	↻	\Righttorque	T	\Tsteel
	\Fixedbearing		\Loosebearing	T	\RoundedLsteel	I	\TTsteel
-	\Flatsteel	L	\Lsteel	L	\RoundedTsteel		

表 116 wasysym 的生物符号

♀	\Female	♂	\FemaleMale	♂	\MALE	○	\Neutral
♀	\FEMALE	♀	\Hermaphrodite	♂	\Male		
♀	\FemaleFemale	♀	\HERMAPHRODITE	♂	\MaleMale		

表 117 marvosym 的生物符号

♀	\Female	♀	\FemaleMale	♂	\MALE	○	\Neutral
♀	\FEMALE	♀	\Hermaphrodite	♂	\Male		
♀	\FemaleFemale	♀	\HERMAPHRODITE	♂	\MaleMale		

表 118 marvosym 的安全符号

☠	\Biohazard	☢	\CEsign	☣	\Explosionsafe	☤	\Radioactivity
☞	\BSEfree	☜	\Estatically	☞	\Laserbeam	☞	\Stopsign

第四部分 Dingbats

表 119 bbding 箭头

➡	\ArrowBoldDownRight	➡	\ArrowBoldRightStrobe
➡	\ArrowBoldRightCircled	➡	\ArrowBoldUpRight
➡	\ArrowBoldRightShort		

表 120 pifont 箭头

➡	\ding{212}	➡	\ding{221}	➡	\ding{230}	➡	\ding{239}	➡	\ding{249}
➡	\ding{213}	➡	\ding{222}	➡	\ding{231}	➡	\ding{241}	➡	\ding{250}
➡	\ding{214}	➡	\ding{223}	➡	\ding{232}	➡	\ding{242}	➡	\ding{251}
➡	\ding{215}	➡	\ding{224}	➡	\ding{233}	➡	\ding{243}	➡	\ding{252}
➡	\ding{216}	➡	\ding{225}	➡	\ding{234}	➡	\ding{244}	➡	\ding{253}
➡	\ding{217}	➡	\ding{226}	➡	\ding{235}	➡	\ding{245}	➡	\ding{254}
➡	\ding{218}	➡	\ding{227}	➡	\ding{236}	➡	\ding{246}		
➡	\ding{219}	➡	\ding{228}	➡	\ding{237}	➡	\ding{247}		
➡	\ding{220}	➡	\ding{229}	➡	\ding{238}	➡	\ding{248}		

表 121 marvosym 剪刀

✂	\Cutleft	✂	\Cutright	✂	\Leftscissors
✂	\Cutline	✂	\Kutline	✂	\Rightscissors

表 122 bbding 剪刀

✂	\ScissorHollowLeft	✂	\ScissorLeftBrokenTop
✂	\ScissorHollowRight	✂	\ScissorRight
✂	\ScissorLeft	✂	\ScissorRightBrokenBottom
✂	\ScissorLeftBrokenBottom	✂	\ScissorRightBrokenTop

表 123 pifont 剪刀





 `\ding{33}`  `\ding{34}`  `\ding{35}`  `\ding{36}`

表 124 dingbat 铅笔

`\largepencil``\smallpencil`

表 125 bbding 铅笔与笔尖











 <code>\NibLeft</code>	 <code>\PencilLeft</code>	 <code>\PencilRightDown</code>
 <code>\NibRight</code>	 <code>\PencilLeftDown</code>	 <code>\PencilRightUp</code>
 <code>\NibSolidLeft</code>	 <code>\PencilLeftUp</code>	
 <code>\NibSolidRight</code>	 <code>\PencilRight</code>	

表 126 pifont 铅笔与笔尖






 `\ding{46}`  `\ding{47}`  `\ding{48}`  `\ding{49}`  `\ding{50}`

表 127 dingbat 手势符号








 <code>\leftpointright</code>	 <code>\rightpointleft</code>	 <code>\rightpointright</code>
 <code>\leftthumbsdown</code>	 <code>\rightthumbsdown</code>	
 <code>\leftthumbsup</code>	 <code>\rightthumbsup</code>	

表 128 bbding 手势符号










 <code>\HandCuffLeft</code>	 <code>\HandCuffRightUp</code>	 <code>\HandPencilLeft</code>
 <code>\HandCuffLeftUp</code>	 <code>\HandLeft</code>	 <code>\HandRight</code>
 <code>\HandCuffRight</code>	 <code>\HandLeftUp</code>	 <code>\HandRightUp</code>

表 129 pifont 手势符号





 `\ding{42}`  `\ding{43}`  `\ding{44}`  `\ding{45}`

表 130 bbding 十字

† \Cross	† \CrossOpenShadow	⊕ \PlusOutline
† \CrossBoldOutline	† \CrossOutline	⊕ \PlusThinCenterOpen
⊕ \CrossClowerTips	⊕ \Plus	
⊕ \CrossMaltese	⊕ \PlusCenterOpen	

表 131 pifont 十字

⊕ \ding{57}	⊕ \ding{59}	† \ding{61}	† \ding{63}
⊕ \ding{58}	⊕ \ding{60}	† \ding{62}	⊕ \ding{64}

表 132 bbding 对错记号

✓ \Checkmark	✕ \XSolid	✕ \XSolidBrush
✓ \CheckmarkBold	✕ \XSolidBold	

表 133 pifont 对错记号

✓ \ding{51}	✕ \ding{53}	✕ \ding{55}
✓ \ding{52}	✕ \ding{54}	✕ \ding{56}

表 134 wasysym 对错记号

☑ \CheckedBox	□ \Square	☒ \XBox
---------------	-----------	---------

表 135 pifont 带圈数字

① \ding{172}	① \ding{182}	① \ding{192}	① \ding{202}
② \ding{173}	② \ding{183}	② \ding{193}	② \ding{203}
③ \ding{174}	③ \ding{184}	③ \ding{194}	③ \ding{204}
④ \ding{175}	④ \ding{185}	④ \ding{195}	④ \ding{205}
⑤ \ding{176}	⑤ \ding{186}	⑤ \ding{196}	⑤ \ding{206}
⑥ \ding{177}	⑥ \ding{187}	⑥ \ding{197}	⑥ \ding{207}
⑦ \ding{178}	⑦ \ding{188}	⑦ \ding{198}	⑦ \ding{208}
⑧ \ding{179}	⑧ \ding{189}	⑧ \ding{199}	⑧ \ding{209}
⑨ \ding{180}	⑨ \ding{190}	⑨ \ding{200}	⑨ \ding{210}
⑩ \ding{181}	⑩ \ding{191}	⑩ \ding{201}	⑩ \ding{211}

表 136 wasysym 星号

☆ \davidstar	* \hexstar	* \varhexstar
--------------	------------	---------------

表 137 bbding 星、花

















































	<code>\Asterisk</code>		<code>\FiveStarOutline</code>
	<code>\AsteriskBold</code>		<code>\FiveStarOutlineHeavy</code>
	<code>\AsteriskCenterOpen</code>		<code>\FiveStarShadow</code>
	<code>\AsteriskRoundedEnds</code>		<code>\FourAsterisk</code>
	<code>\AsteriskThin</code>		<code>\FourCloverOpen</code>
	<code>\AsteriskThinCenterOpen</code>		<code>\FourCloverSolid</code>
	<code>\DavidStar</code>		<code>\FourStar</code>
	<code>\DavidStarSolid</code>		<code>\FourStarOpen</code>
	<code>\EightAsterisk</code>		<code>\JackStar</code>
	<code>\EightFlowerPetal</code>		<code>\JackStarBold</code>
	<code>\EightFlowerPetalRemoved</code>		<code>\SixFlowerAlternate</code>
	<code>\EightStar</code>		<code>\SixFlowerAltPetal</code>
	<code>\EightStarBold</code>		<code>\SixFlowerOpenCenter</code>
	<code>\EightStarConvex</code>		<code>\SixFlowerPetalDotted</code>
	<code>\EightStarTaper</code>		<code>\SixFlowerPetalRemoved</code>
	<code>\FiveFlowerOpen</code>		<code>\SixFlowerRemovedOpenPetal</code>
	<code>\FiveFlowerPetal</code>		<code>\SixStar</code>
	<code>\FiveStar</code>		<code>\SixteenStarLight</code>
	<code>\FiveStarCenterOpen</code>		<code>\Snowflake</code>
	<code>\FiveStarConvex</code>		<code>\SnowflakeChevron</code>
	<code>\FiveStarLines</code>		<code>\SnowflakeChevronBold</code>
	<code>\FiveStarOpen</code>		<code>\Sparkle</code>
	<code>\FiveStarOpenCircled</code>		<code>\SparkleBold</code>
	<code>\FiveStarOpenDotted</code>		<code>\TwelveStar</code>

表 138 pifont 星、花

















































































































	<code>\ding{65}</code>		<code>\ding{74}</code>		<code>\ding{83}</code>		<code>\ding{92}</code>		<code>\ding{101}</code>
	<code>\ding{66}</code>		<code>\ding{75}</code>		<code>\ding{84}</code>		<code>\ding{93}</code>		<code>\ding{102}</code>
	<code>\ding{67}</code>		<code>\ding{76}</code>		<code>\ding{85}</code>		<code>\ding{94}</code>		<code>\ding{103}</code>
	<code>\ding{68}</code>		<code>\ding{77}</code>		<code>\ding{86}</code>		<code>\ding{95}</code>		<code>\ding{104}</code>
	<code>\ding{69}</code>		<code>\ding{78}</code>		<code>\ding{87}</code>		<code>\ding{96}</code>		<code>\ding{105}</code>
	<code>\ding{70}</code>		<code>\ding{79}</code>		<code>\ding{88}</code>		<code>\ding{97}</code>		<code>\ding{106}</code>
	<code>\ding{71}</code>		<code>\ding{80}</code>		<code>\ding{89}</code>		<code>\ding{98}</code>		<code>\ding{107}</code>
	<code>\ding{72}</code>		<code>\ding{81}</code>		<code>\ding{90}</code>		<code>\ding{99}</code>		
	<code>\ding{73}</code>		<code>\ding{82}</code>		<code>\ding{91}</code>		<code>\ding{100}</code>		

表 139 wasysym 几何形状

	<code>\hexagon</code>		<code>\octagon</code>		<code>\pentagon</code>		<code>\varhexagon</code>
---	-----------------------	---	-----------------------	---	------------------------	---	--------------------------

表 140 ifsym 几何形状

	<code>\BigCircle</code>		<code>\FilledSmallTriangleUp</code>
	<code>\BigCross</code>		<code>\FilledSquare</code>
	<code>\BigDiamondshape</code>		<code>\FilledSquareShadowA</code>
	<code>\BigHBar</code>		<code>\FilledSquareShadowC</code>
	<code>\BigLowerDiamond</code>		<code>\FilledTriangleDown</code>
	<code>\BigRightDiamond</code>		<code>\FilledTriangleLeft</code>
	<code>\BigSquare</code>		<code>\FilledTriangleRight</code>
	<code>\BigTriangleDown</code>		<code>\FilledTriangleUp</code>
	<code>\BigTriangleLeft</code>		<code>\HBar</code>
	<code>\BigTriangleRight</code>		<code>\LowerDiamond</code>
	<code>\BigTriangleUp</code>		<code>\RightDiamond</code>
	<code>\BigVBar</code>		<code>\SmallCircle</code>
	<code>\Circle</code>		<code>\SmallCross</code>
	<code>\Cross</code>		<code>\SmallDiamondshape</code>
	<code>\DiamondShadowA</code>		<code>\SmallHBar</code>
	<code>\DiamondShadowB</code>		<code>\SmallLowerDiamond</code>
	<code>\DiamondShadowC</code>		<code>\SmallRightDiamond</code>
	<code>\Diamondshape</code>		<code>\SmallSquare</code>
	<code>\FilledBigCircle</code>		<code>\SmallTriangleDown</code>
	<code>\FilledBigDiamondshape</code>		<code>\SmallTriangleLeft</code>
	<code>\FilledBigSquare</code>		<code>\SmallTriangleRight</code>
	<code>\FilledBigTriangleDown</code>		<code>\SmallTriangleUp</code>
	<code>\FilledBigTriangleLeft</code>		<code>\SmallVBar</code>
	<code>\FilledBigTriangleRight</code>		<code>\SpinDown</code>
	<code>\FilledBigTriangleUp</code>		<code>\SpinUp</code>
	<code>\FilledCircle</code>		<code>\Square</code>
	<code>\FilledDiamondShadowA</code>		<code>\SquareShadowA</code>
	<code>\FilledDiamondShadowC</code>		<code>\SquareShadowB</code>
	<code>\FilledDiamondshape</code>		<code>\SquareShadowC</code>
	<code>\FilledSmallCircle</code>		<code>\TriangleDown</code>
	<code>\FilledSmallDiamondshape</code>		<code>\TriangleLeft</code>
	<code>\FilledSmallSquare</code>		<code>\TriangleRight</code>
	<code>\FilledSmallTriangleDown</code>		<code>\TriangleUp</code>
	<code>\FilledSmallTriangleLeft</code>		<code>\VBar</code>
	<code>\FilledSmallTriangleRight</code>		

为使用本组符号, 必须在调入 ifsym 时使用选项 `geometry`. 还可以利用 `\rlap` 生成新的符号, 例如: `\rlap\FilledSmallCircle\BigCircle` 可以生成大圈套小圆, `\rlap\ifsCross\ifsSquare` 可以产生方框打叉等.

表 141 bbding 几何形状






















	<code>\CircleShadow</code>		<code>\Square</code>
	<code>\CircleSolid</code>		<code>\SquareCastShadowBottomRight</code>
	<code>\DiamondSolid</code>		<code>\SquareCastShadowTopLeft</code>
	<code>\Ellipse</code>		<code>\SquareCastShadowTopRight</code>
	<code>\EllipseShadow</code>		<code>\SquareShadowBottomRight</code>
	<code>\EllipseSolid</code>		<code>\SquareShadowTopLeft</code>
	<code>\HalfCircleLeft</code>		<code>\SquareShadowTopRight</code>
	<code>\HalfCircleRight</code>		<code>\SquareSolid</code>
	<code>\Rectangle</code>		<code>\TriangleDown</code>
	<code>\RectangleBold</code>		<code>\TriangleUp</code>
	<code>\RectangleThin</code>		

表 142 pifont 几何形状















	<code>\ding{108}</code>		<code>\ding{111}</code>		<code>\ding{114}</code>		<code>\ding{117}</code>		<code>\ding{121}</code>
	<code>\ding{109}</code>		<code>\ding{112}</code>		<code>\ding{115}</code>		<code>\ding{119}</code>		<code>\ding{122}</code>
	<code>\ding{110}</code>		<code>\ding{113}</code>		<code>\ding{116}</code>		<code>\ding{120}</code>		

表 143 universe 几何形状







	<code>\baucircle</code>		<code>\bausquare</code>		<code>\bautriangle</code>
--	-------------------------	--	-------------------------	--	---------------------------

表 144 manfnt 的危险转折符号

	<code>\dbend</code>		<code>\lhdbend</code>		<code>\reversedvideobend</code>
---	---------------------	---	-----------------------	---	---------------------------------

请注意这些符号都延伸到基线下面，manfnt 也定义了不下沉的符号，分别称为：
`\textdbend`, `\textlhdbend`, `\textreversedvideobend`.

表 145 marvosym 消息符号














	<code>\Bicycle</code>		<code>\Football</code>		<code>\Pointinghand</code>
	<code>\Checkedbox</code>		<code>\Gentsroom</code>		<code>\Wheelchair</code>
	<code>\Clocklogo</code>		<code>\Industry</code>		<code>\Writinghand</code>
	<code>\Coffeecup</code>		<code>\Info</code>		
	<code>\Crossedbox</code>		<code>\Ladiesroom</code>		

表 146 skull 符号


	<code>\skull</code>
---	---------------------

表 147 mathabx 的非数学符号

‡ \rip

表 148 dingbat 其他符号

⚓ \anchor	👁 \eye	▤ \Sborder
↩ \carriagereturn	⬛ \filledsquarewithdots	⊠ \squarewithdots
✓ \checkmark	📶 \satellitedish	▤ \Zborder

表 149 bbding 其他符号

✉ \Envelope	☎ \Phone	☀ \SunshineOpenCircled
❖ \OrnamentDiamondSolid	📞 \PhoneHandset	📼 \Tape
☺ \Peace	✈ \Plane	

表 150 pifont 其他符号

☎ \ding{37}	✉ \ding{41}	☀ \ding{166}	♥ \ding{170}
📞 \ding{38}	❖ \ding{118}	📼 \ding{167}	♠ \ding{171}
📼 \ding{39}	♥ \ding{164}	♣ \ding{168}	
✈ \ding{40}	📶 \ding{165}	♦ \ding{169}	

第五部分 其他符号

表 151 textcomp 宗谱符号

★ \textborn	∘ \textdivorced	∞ \textmarried
† \textdied	🌿 \textleaf	

表 152 wasysym 一般符号

☒ \ataribox	🕒 \clock	◀ \LEFTarrow	☺ \smiley
🔔 \bell	∅ \diameter	⚡ \lightning	☀ \sun
👤 \blacksmiley	▼ \DOWNarrow	☎ \phone	▲ \UParrow
🎩 \Bowtie	☹ \frownie	☞ \pointer	◻ \wasylzenge
! \brokenvert	⊗ \invdiameter	📻 \recorder	
✓ \checked	✝ \kreuz	▶ \RIGHTarrow	

表 153 wasysym 圆周

● \CIRCLE	◐ \LEFTcircle	◑ \RIGHTcircle	↻ \rightturn
○ \Circle	◑ \Leftcircle	◐ \Rightcircle	
◑ \LEFTCIRCLE	◐ \RIGHTCIRCLE	↻ \leftturn	

表 154 wasysym 音符

♪ \eighthnote	♩ \twonotes	♩ \quarternote
♩ \halfnote	。 \fullnote	

表 155 harmony 音符

♪ \AAcht	§ \Ds	· \Pu	> \VM
♩ \Acht	§ \DS	♩ \Sech	♩ \Zwdr
7 \AcPa	o \Ganz	7 \SePa	7 \ZwPa
♩ \DD	- \GaPa	< \UB	
♩ \DDohne	♩ \Halb	♩ \Vier	
♩ \Dohne	- \HaPa	z \ViPa	

必须读入 musixtex 才能使用 harmony.

表 156 manfnt 的其他符号

◐ \manboldkidney	◐ \manpenkidney
◎ \manconcentriccircles	☼ \manquadrifolium
◈ \manconcentricdiamond	↷ \manquartercircle
◊ \mancone	☼ \manrotatedquadrifolium
◩ \mancube	↷ \manrotatedquartercircle
↗ \manerrarrow	☆ \manstar
◐ \manfilledquartercircle	↗ \mantilt pennib
— \manhpennib	▼ \mantriangledown
◩ \manimpossiblecube	► \mantriangleright
◐ \mankidney	▲ \mantriangleup
◐ \manlhpenkidney	! \manvpennib

表 157 marvosym 的导航符号

► \Forward	▼ \MoveDown	◀◀ \RewindToIndex	▲ \ToTop
►► \ForwardToEnd	▲ \MoveUp	◀◀ \RewindToStart	
►►► \ForwardToIndex	◀ \Rewind	▼ \ToBottom	

表 158 marvosym 的熨烫符号















































 \AtForty	 \Handwash	 \ShortNinetyFive
 \AtNinetyFive	 \IroningI	 \ShortSixty
 \AtSixty	 \IroningII	 \ShortThirty
 \Bleech	 \IroningIII	 \SpecialForty
 \CleaningA	 \NoBleech	 \Tumbler
 \CleaningF	 \NoChemicalCleaning	 \WashCotton
 \CleaningFF	 \NoIroning	 \WashSynthetics
 \CleaningP	 \NoTumbler	 \WashWool
 \CleaningPP	 \ShortFifty	
 \Dontwash	 \ShortForty	

表 159 marvosym 的其他符号






















 \Ankh	 \Cross	 \Heart	 \Smiley
 \Bat	 \FHBLogo	 \MartinVogel	 \Womanface
 \Bouquet	 \FHBLogo	 \Mundus	 \Yinyang
 \Celtcross	 \Frowny	 \MVAt	
 \CircledA	 \FullFHB	 \Rightarrow*	

* 注意这里的定义与 L^AT_EX 不同!

表 160 universa 的其他符号








 \bauforms	 \bauhead
---	--

表 161 ifsym 的气象符号

 \Cloud	 \Hail	 \Snow
 \FilledCloud	 \HalfSun	 \SnowCloud
 \FilledRainCloud	 \Lightning	 \Sun
 \FilledSnowCloud	 \NoSun	 \SunCloud
 \FilledSunCloud	 \Rain	 \ThinFog
 \FilledWeakRainCloud	 \RainCloud	 \WeakRain
 \Fog	 \Sleet	 \WeakRainCloud


















为使用本组符号, 必须在调入 ifsym 时使用选项 weather.

表 162 ifsym 的时钟符号

 \Interval	 \StopWatchStart	 \VarClock	 \Wecker
 \StopWatchEnd	 \Taschenuhr	 \VarTaschenuhr	















为使用本组符号, 必须在调入 ifsym 时使用选项 clock.

表 163 ifsym 的登山符号

	<code>\FilledHut</code>		<code>\Mountain</code>		<code>\VarFlag</code>
	<code>\Flag</code>		<code>\StoneMan</code>		<code>\VarIceMountain</code>
	<code>\HalfFilledHut</code>		<code>\Summit</code>		<code>\VarMountain</code>
	<code>\Hut</code>		<code>\SummitSign</code>		<code>\VarSummit</code>
	<code>\IceMountain</code>		<code>\SurveySign</code>		<code>\Village</code>
	<code>\Joch</code>		<code>\Tent</code>		















为使用本组符号, 必须在调入 ifsym 时使用选项 alpine.

表 164 ifsym 的其他符号

	<code>\FilledSectioningDiamond</code>		<code>\Letter</code>		<code>\Radiation</code>
	<code>\Fire</code>		<code>\PaperLandscape</code>		<code>\SectioningDiamond</code>
	<code>\Irritant</code>		<code>\PaperPortrait</code>		<code>\Telephone</code>
	<code>\StrokeOne</code>		<code>\StrokeThree</code>		<code>\StrokeFive</code>
	<code>\StrokeTwo</code>		<code>\StrokeFour</code>		

为使用本组符号, 必须在调入 ifsym 时使用选项 misc.

表 165 dictsym 的辞书符号

	<code>\dsaeronautical</code>		<code>\dscommercial</code>		<code>\dsmedical</code>
	<code>\dsagricultural</code>		<code>\ds heraldical</code>		<code>\dsmilitary</code>
	<code>\dsarchitectural</code>		<code>\dsjuridical</code>		<code>\dsrailways</code>
	<code>\dsbiological</code>		<code>\dsliterary</code>		<code>\dstechnical</code>
	<code>\dschemical</code>		<code>\ds mathematical</code>		

参考文献与网站

1. Knuth D. E. *The T_EXbook*. NJ: Addison-Wesley, 1984
2. Knuth D. E. *The METAFONTbook*. NJ: Addison-Wesley, 1986
3. Kopka H., Daly P. W. *A Guide to L^AT_EX*. Third edition. NJ: Addison-Wesley, 1999
4. Lamport L. *L^AT_EX—A Document Preparation System*. NJ: Addison-Wesley, 1985
5. Salomon D. *The Advanced T_EXbook*. Berlin: Springer-Verlag, 1995
6. Seroul R., Levy S. *A Beginner's Book of T_EX*. NJ: Springer-Verlag, 1991
7. Spivak M. D. *The Joy of T_EX*. Second edition. American Mathematical Society, 1990
8. von Bechtolsheim S. *T_EX in Practice*. vol. I – IV. Berlin: Springer-Verlag, 1993
9. 张林波: 关于新版 CCT 的说明, <ftp://ftp.cc.ac.cn/pub/cct/README.pdf>, 2003

国内有关 T_EX 的网站有:

<http://www.ctex.org>

<http://www.chinatex.org>

国外的网站有:

<http://www.tug.org>

这是 T_EX 用户协会 (T_EX Users Group) 的主页, 有各种最新消息以及 T_EX 有关材料的下载.

要下载 T_EX 的各种最新版本, 可到 CTAN (Comprehensive T_EX Archive Network) 用匿名 ftp 下载, CTAN 站点的地址是:

<ftp://cam.ctan.org/tex-archive/> (英国)

<ftp://dante.ctan.org/tex-archive/> (德国)

<ftp://tug.ctan.org/tex-archive/> (美国)

此外, 以下站点的 'L^AT_EX 百科全书' 也是值得一看的地方:

<http://tex.loria.fr/english/index.html>